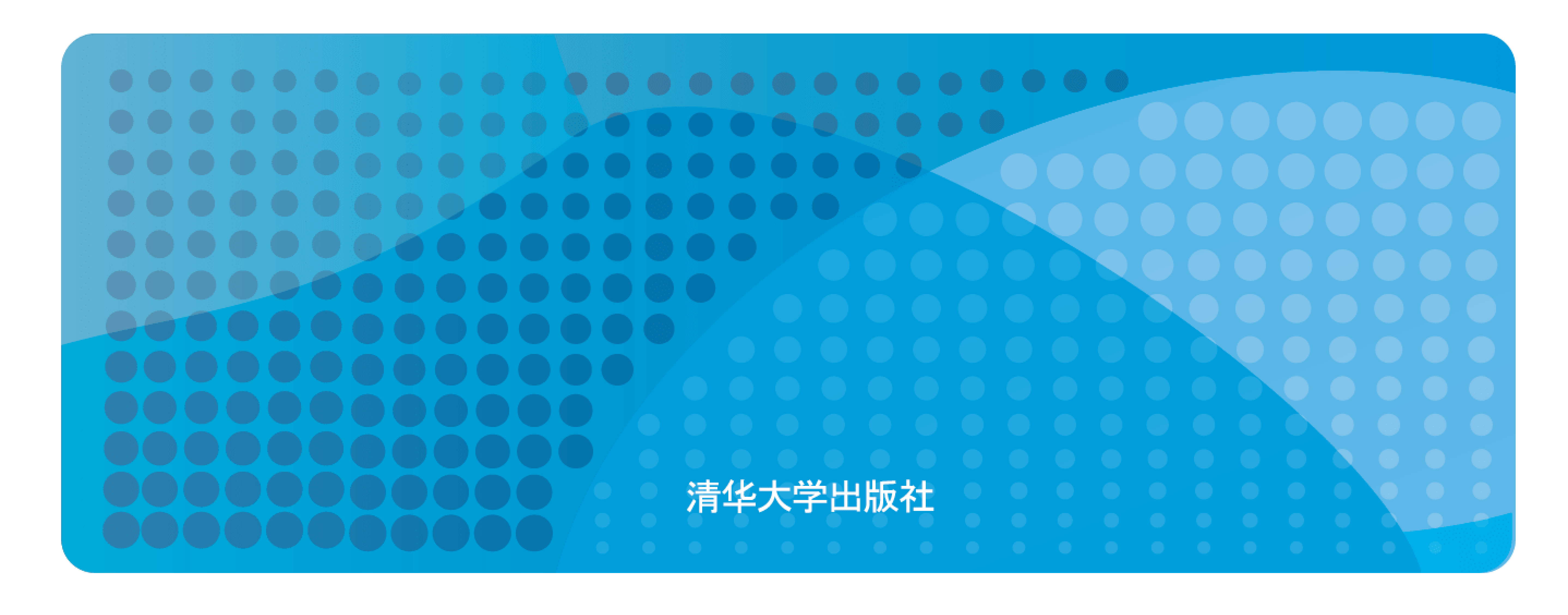


ASP.NET
动态网站设计与制作

关　忠　于洪霞　主　编
王　耀　李　毅　副主编



应用型大学计算机专业系列教材

ASP.NET动态网站设计与制作

关　忠　于洪霞　主　编
王　耀　李　毅　副主编

清华大学出版社
北　京

内 容 简 介

本书重点介绍基于 ASP.NET(C＃语言)技术的动态网站的制作与实现，主要内容包括网站开发基础、C＃、ASP.NET、数据库技术、网站设计、网站快速开发技术、网站建设实例，并通过实例指导学生实训，加强实践，强化技能培养。

本书知识系统、案例丰富、语言简洁、突出实用性，便于学习掌握，既可作为应用型大学本科及高职高专院校信息管理、计算机应用、网络管理、电子商务等专业教学的教材，也可用于广大企事业单位 IT 从业人员的职业教育和在职培训，并可为网站设计爱好者和程序员实际工作提供有益的参考。

图书在版编目(CIP)数据

ASP.NET 动态网站设计与制作/关忠，于洪霞主编.--北京：清华大学出版社，2016
应用型大学计算机专业系列教材
ISBN 978-7-302-42904-3

Ⅰ.①A… Ⅱ.①关… ②于… Ⅲ.①网页制作工具－程序设计－高等学校－教材
Ⅳ.①TP393.092

中国版本图书馆 CIP 数据核字(2016)第 030102 号

责任编辑：王剑乔
封面设计：
责任校对：李　梅
责任印制：

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62795764
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　**印　张：**16　　**字　　数：**363 千字
版　　次：2016 年 3 月第 1 版　　**印　　次：**2016 年 3 月第1 次印刷
印　　数：1～　000
定　　价：　.00 元

产品编号：067756-01

编审委员会

序言

PREFACE

微电子技术、计算机技术、网络技术、通信技术、多媒体技术等高新科技日新月异的飞速发展和普及应用，不仅有力地促进了各国经济发展、加速了全球经济一体化的进程，而且促进着当今世界迅速跨入信息社会。以计算机为主导的计算机文化，正在深刻地影响人类社会的经济发展与文明建设；以网络为基础的网络经济，正在全面地改变传统的社会生活、工作方式和商务模式。当今社会，计算机应用水平、信息化发展速度与程度，已经成为衡量一个国家经济发展和竞争力的重要指标。

目前我国正处于经济快速发展与社会变革的重要时期，随着经济转型、产业结构调整、传统企业改造，涌现了大批电子商务、新媒体、动漫、艺术设计等新型文化创意产业，而这一切都离不开计算机，都需要网络等现代化信息技术手段的支撑。处于网络时代、信息化社会，今天人们所有工作都已经全面实现了计算机化、网络化，当今更加强调计算机应用与行业、企业的结合，更注重计算机应用与本职工作、具体业务的紧密结合。当前，面对国际市场的激烈竞争和巨大的就业压力，无论是企业还是即将毕业的学生，掌握好计算机应用技术已成为求生存、谋发展的关键技能。

没有计算机就没有现代化！没有计算机网络就没有我国经济的大发展！为此，国家出台了一系列关于加强计算机应用和推动国民经济信息化进程的文件及规定，启动了"电子商务、电子政务、金税"等具有深刻含义的重大工程，加速推进"国防信息化、金融信息化、财税信息化、企业信息化、教育信息化、社会管理信息化"，全社会又掀起新一轮计算机学习应用的热潮，因此，本套教材的出版具有特殊意义。

针对我国应用型大学"计算机应用"等专业知识老化、教材陈旧、重理论轻实践、缺乏实际操作技能训练等问题，为了适应我国国民经济信息化发展对计算机应用人才的需要，为了全面贯彻教育部关于"加强职业教育"精神和"强化实践实训、突出技能培养"的要求，根据企业用人与就业岗位的真实需要，结合应用型大学"计算机应用"和"网络管理"等专业的教学计划及课程设置与调整的实际情况，我们组织北京联合大学、陕西理工学院、北方工业大学、华北科技学院、北京财贸职业学院、山东滨州职业学院、山西大学、首钢工学院、包头职业技术学院、北京科技大学、广东理工学院、北京城市学院、郑州大学、北京朝阳社区学院、哈尔滨师范大学、黑龙江工商大学、北京石景山社区学院、海南职业学院、北京西城经济科学大学等全国 30 多所高校及高职院校的计算机教师和具有丰富实践经验的企业人士共同撰写了此套教材。

本套教材包括《ASP. NET 动态网站设计与制作》《数据库技术应用教程(SQL Server 2012 版)》《Web 静态网页设计与排版》《中小企业网站建设与管理》等。在编写过程中，全

体作者注意坚持以科学发展观为统领，严守统一的创新型案例教学格式化设计，采取任务制或项目制写法；注重校企结合，贴近行业企业岗位实际，注重实用性技术与应用能力的训练培养，注重实践技能应用与工作背景紧密结合，同时也注重计算机、网络、通信、多媒体等现代化信息技术的新发展，具有集成性、系统性、针对性、实用性、易于实施教学等特点。

本套教材不仅适合应用型大学及高职高专院校计算机应用、网络、电子商务等专业学生的学历教育，同时也可作为工商、外贸、流通等企事业单位从业人员的职业教育和在职培训，对于广大社会自学者也是有益的参考学习读物。

系列教材编委会

2015 年 10 月

前言

FOREWORD

网络经济推动国民经济快速发展，企业网站运营作为现代科技进步催生的新型生产力，不仅在拉动内需、解决就业、扩大经营、促进经济发展、加速传统产业升级、提高企业竞争力等方面发挥着重要作用，而且也在彻底改造企业的经营管理方式，并在深刻地改变着企业商务活动的运作模式，因而越来越受到各级政府和各类企业的重视。

随着计算机与网络通信技术的飞速发展，计算机网络应用已经渗透到社会经济领域的各个方面。企业网站建设既是信息化推进的基础，也是网络经济发展的关键环节。

动态网站设计制作是高等院校计算机应用和网络管理专业重要的核心课程，也是计算机网络及软件相关专业中常设的一门专业课；当前学习动态网站设计制作知识、掌握动态网站设计开发应用的关键技能，已经成为网站及 IT 从业者的先决和必要条件。

目前我国正处于经济改革与社会发展的关键时期，随着国民经济信息化的迅猛发展，面对 IT 市场的激烈竞争、面对就业的巨大压力，努力学好动态网站设计制作，真正掌握现代化网络开发工具，对于今后的发展具有特殊的意义。

本书作为高等教育应用型大学本科及高职高专院校计算机应用和网络管理专业的特色教材，全书共 7 章。以学习者应用能力培养为主线，坚持科学发展观，严格按照国家教育部关于"加强职业教育、突出实践技能培养"的要求，根据应用型大学教学改革的需要，依照动态网站设计制作学习和应用的基本过程和规律，采用"任务驱动、案例教学"写法，突出"实例与理论的紧密结合"，循序渐进地进行知识要点的讲解。

本书的特色和价值如下。

1. 项目驱动，案例教学，面向岗位，突出技能培养。每一章都包含丰富的案例，让读者可以理论联系实际，既掌握了基础知识的原理，也掌握了如何运用这些基础知识的方法。

2. 完善学生的知识结构，培养复合型人才，提升学生就业竞争力。本书从动态网站开发的各个角度进行了讲解，让学生完整地掌握使用 ASP. NET 技术开发动态网站的方法，帮助学生掌握独立开发动态网站的能力。

3. 提供方便老师和学生使用的辅助资源包。其主要包括书中每一个章节使用的完整的程序源代码文件，以及相关辅助文件(数据库文件、资源文件等)，读者在学习书中内容的同时，可以运行这些实例，更深入地掌握和领会书中内容。

本书融入了最新的实践教学理念，力求严谨，注重与时俱进，具有知识系统、案例丰富、贴近实际、突出实用性等特点。本书既可作为应用型大学本科及高职高专院校计算机应用、网络管理、电子商务等专业教学的教材，也可用于广大企事业单位 IT 从业人员的

职业教育和在职培训，并可为网站设计爱好者和程序员实际工作提供有益的参考。

本书由李大军筹划并具体组织，关忠和于洪霞主编，关忠统改稿，王耀、李毅为副主编，由具有丰富动态网站设计制作教学与实践经验的邵晶波（博士）教授审订。作者编写分工为牟惟仲编写序言，李妍编写第 1 章，李毅编写第 2 章，于洪霞编写第 3 章和第 7 章，刘靖宇编写第 4 章和附录，关忠编写第 5 章，王耀编写第 6 章，由华燕萍、李晓新进行文字修改、版式调整并制作教学课件。

在本书编写过程中，参阅借鉴了大量国内外有关动态网站制作的最新书刊、相关网站资料、国家历年颁布实施的法规和管理规定，并得到编委会及业界专家教授的具体指导，在此一并致谢。为配合本书的发行使用，提供配套电子课件和资源包，读者可以从清华大学出版社网站（www. tup. com. cn）免费下载。

因动态网站设计制作技术发展较快且作者水平有限，书中难免存在疏漏和不足，恳请同行和读者批评指正。

编　者

2016 年 1 月

目录

CONTENTS

第 1 章

网站开发基础

随着 Internet 的迅猛发展，网络正极大地改变着人类的生活和工作方式。通过网络可以购物、搜索信息、看病挂号、预订各项服务等，这一切都离不开 Web 网站的支撑。在开始正式学习动态网站开发之前，要先对动态网站开发有一个基本的了解。

本章将介绍开发网站的一些基础知识，通过本章的学习，将使读者了解网页设计的基本概念、构成网页的基本要素、制作网页的基本工具和网站运行的环境。

1.1 网站基础知识

Internet 是一个将世界各种不同计算机网络连接起来的全球性网络。网站是一种沟通工具，人们可以通过网站来发布自己想要公开的资讯，或者利用网站来提供相关的网络服务。人们可以通过网页浏览器来访问网站，获取自己需要的资讯或者享受网络服务。

在学习相关知识之前先浏览一些经典网站页面效果，如图 1-1 所示。

1.1.1 WWW 简介

WWW 是环球信息网的缩写，英文全称为 World Wide Web，中文名字为“万维网”“环球网”等，常简称为 Web。

WWW 分为 Web 客户端和 Web 服务器程序。WWW 可以让 Web 客户端(常用浏览器)访问浏览 Web 服务器上的页面。它利用超文本(hypertext)、超媒体(hypermedia)等技术，用户通过浏览器可以方便地通过全局“统一资源标识符”(URI)标识检索远程服务器上的资源(文本、图片、声音及视频文件)，这些资源通过超文本传输协议(HyperText Transfer Protocol，HTTP)传送给用户。

1. 统一资源定位器

统一资源定位器(Uniform Resource Locator，URL)是对可以从互联网上得到的资源的位置和访问方法的一种简洁表示，是互联网上标准资源的地址。互联网上的每个文



(a) Black Angus Steakhouse餐厅官网



(b) 捷豹汽车官网



(c) 游戏官网

图 1-1　经典网站页面效果

件都有一个唯一的 URL,它包含的信息指出文件的位置以及浏览器应该怎么处理它。简单地说,URL 就是 WWW 服务器主机的地址,也称为网址。

URL 的结构是有一定规则的,它的语法格式如下:

协议名称://服务器主机名称[:通信端口/文件目录/文件名称]

例如,http://www.sina.com.cn/news/2015-03-27/08221416883.html,其中 http 是 WWW 服务器与客户机之间遵循的通信协议;www.sina.com.cn 用来标识该文件存储在哪个服务器上;/news/2015-03-27/是文件所在的目录路径;08221416883.html 是这个文件的名称。

2. 超文本标记语言

超文本标记语言(HyperText Markup Language,HTML)是一种专门用于 WWW 的编程语言,用于描述超文本各个部分的构造,告诉浏览器如何显示文本,怎样生成文档链接等信息。使用 HTML 编写的文件之所以被称为超文本文件,是因为它能独立于各种操作系统平台。HTML 文件中可以加入图片、声音、动画、影视等网页内容,并且可以从一个文件跳转到另一个文件,与世界各地主机的文件链接。

1.1.2 网页和网站

1. 网页

网页是用 HTML 或者其他语言编写的,通过 IE 浏览器编译后供用户获取信息的页面,又称为 Web 页,其中可包含文字、图像、表格、动画和超链接等各种网页元素。网页一般可分为静态网页和动态网页。

静态网页是指网页文件中没有程序,而只有 HTML 代码,一般以.html 或者.htm 为后缀名,如图 1-2(a)所示。

动态网页是指网页文件中不仅具有 HTML 标记,而且还含有程序代码,并通过数据库建立连接,通常以.asp、.aspx、.jsp、.php 等为后缀名。这种文档类型的网页由于采用了动态网页技术,所以拥有更好的交互性、安全性和友好性,如图 1-2(b)所示。

2. 网站

网站是指在互联网上,根据一定的规则,使用 HTML 等工具制作的用于展示特定内容的相关网页集合,它建立在网络基础之上,以计算机、网络和通信技术为依托,通过一台或多台计算机向访问者提供服务。平时所说的访问某个站点,实际上访问的是提供这种服务的一台或多台计算机。

网站和网页的区别在于网站是一个整体,而网页是一个个体。网站是有独立域名、独立存放空间的内容集合,这些内容可能是网页,也可能是程序或其他文件,不一定要有很多网页,主要有独立域名和空间,哪怕只有一个页面也叫网站。

网页是网站的组成部分。有了很多网页没有独立的域名和空间也只能说是网页,如 Blog、挂在别人那里的个人主页、网店等尽管有很多页面,功能也齐全,但都不能叫网站。



(a) 静态网页



(b) 动态网页

图 1-2　网页

1.2　静态网页开发语言

静态网页有时也称为平面页，是网站建设的基础，早期的网站一般都是由静态网页制作的。

1.2.1　静态网页

1. 静态网页概述

在网站设计中，静态网页是标准的 HTML 文件，它的文件扩展名是.htm、.html，可以包含文本、图像、声音、Flash 动画、客户端脚本和 ActiveX 控件及 Java 小程序等；可以出现各种动态的效果，如 GIF 格式的动画、Flash、滚动字幕等。这些“动态效果”只是视觉上的，相对于动态网页而言，静态网页主要指没有后台数据库、不含程序和不可交互的网

页。静态网页更新起来相对比较麻烦,适用于一般更新较少的展示型网站。

2. 静态网页的特点

(1) 静态网页每个页面都有一个固定的 URL,且网页 URL 以.htm、.html、.shtml等常见形式为后缀。

(2) 静态网页是实实在在保存在服务器上的文件,每个网页都是一个独立的文件。

(3) 静态网页的内容相对稳定,因此容易被搜索引擎检索。

(4) 静态网页没有数据库,在网站制作和维护方面工作量较大。

(5) 静态网页的交互性较差,在功能方面有较大的限制。

(6) 页面浏览迅速,开启页面速度快于动态页面。

(7) 减轻了服务器的负担,工作量减少,也降低了数据库的成本。

1.2.2　HTML 的基本知识

1. HTML 的基本概念

超文本标记语言是为网页创建和其他可在网页浏览器中看到的信息设计的一种标记语言。超文本标记语言是标准通用标记语言下的一个应用,也是一种规范、一种标准,它通过标记符号来标记要显示的网页中的各个部分。

网页文件本身是一种文本文件,通过在文本文件中添加标记符,可以告诉浏览器如何显示其中的内容(如文字如何处理、画面如何安排、图片如何显示等)。

浏览器按顺序阅读网页文件,然后根据标记符解释和显示其标记的内容,对书写出错的标记将不指出其错误,且不停止其解释执行过程,编制者只能通过显示效果来分析出错原因和出错部位。需要注意的是,对于不同的浏览器,对同一标记符可能会有不完全相同的解释,因而可能会有不同的显示效果。

2. HTML 语言的工作原理

HTML 编程就是用各种各样的标记把在页面中要显示的内容组织起来,其中包含了许多标记和文本,浏览器可以理解这些标记,并把标记解译出来,显示出所浏览的网页。

3. 基本的 HTML 语法

在 HTML 语言中,所有标记都须用<>括起来,如<HTML>、<HEAD>、<BODY>等。

大部分标记都是成对出现的,包括开始标记和结束标记,开始标记和结束标记定义了所影响的范围。结束标记与开始标记名称相同,但结束标记总是以一个斜线符号开头的,例如,<HTML>和</HTML>。也有一些标记没有结束标记符号,如换行标记
。

标记可放置在网页中的任意部位,浏览器不会把这些标记本身显示出来,只是按照标记的要求对标记符之间的内容进行特殊显示。每种标记有不同的作用,学习 HTML 语言就是学习这些各种各样的标记。

4. HTML 的基本结构

HTML 文档主要由 3 部分组成，如图 1-3 所示。

```
<HTML>
  <HEAD>
    <TITLE>标题文字</TITLE>
  <HEAD>
  <BODY>
      文本、图像、动画等
  </BODY>
</HTML>
```

图 1-3 HTML 的基本结构

1）HTML 部分

HTML 部分以＜HTML＞标签开始，以＜/HTML＞标签结束。



【基本语法】

```
<HTML>
⋮
</HTML>
```

＜HTML＞标签告诉浏览器这两个标签中间的内容是 HTML 文档。

2）头部

头部以＜HEAD＞标签开始，以＜/HEAD＞标签结束。这部分包含显示在网页标题栏中的标题和其他在网页中不显示的信息。标题包含在＜TITLE＞和＜/TITLE＞标签之间。



【基本语法】

```
<HEAD>
<TITLE>…</TITLE>
</HEAD>
```

3）主体部分

主体部分包含在网页中显示的文本、图像和链接。主体部分以＜BODY＞标签开始，以＜/BODY＞标签结束。



【基本语法】

```
<BODY>
⋮
</BODY>
```



【小提示】

标签不区分大小写，因此用户可以使用＜html＞来代替＜HTML＞，其他标记雷同。

5. 标记符的属性

大多数标记都包含有多个属性，通过这些属性可以对作用的内容进行更多地控制。在 HTML 语言中，所有属性都放置在开始标记的尖括号内。



【基本语法】

```
<标记名  属性名=属性值  属性名=属性值>文本</标记名>
```

示例代码如下。

```
<font face = "宋体" size = "2">静态网页</font>
```

说明：标记名为 font。font 的属性有 face 和 size。face 属性值为“宋体”，size 属性值为 2。

1.2.3　HTML 常用标记

1. 标题标记<hn>

标题标记<hn>...</hn>用于设置文档中的标题。标题标签中的 n 代表 1～6 这个范围，<h1>定义最大的标题，<h6>定义最小的标题。

示例代码如下。

```
<html>
<body>
<h1>一级标题</h1>
<h2>二级标题</h2>
<h3>三级标题</h3>
<h4>四级标题</h4>
<h5>五级标题</h5>
<h6>六级标题</h6>
</body>
</html>
```

具体效果如图 1-4 所示。



图 1-4　hn 标记效果

2. 字体标记

字体标记...用来设置文本的字符格式，将文本置于和标记之间，通过属性 face、size 和 color 来设置字体、字号和颜色。

示例代码如下。

```
<html>
<body>
<h1><font face = "隶书">文字设置</font></h1>
<p><font size = "7" face = "楷体" color = "red">设置文字的字体、颜色和大小</font></p>
</body>
</html>
```

设置效果如图 1-5 所示。

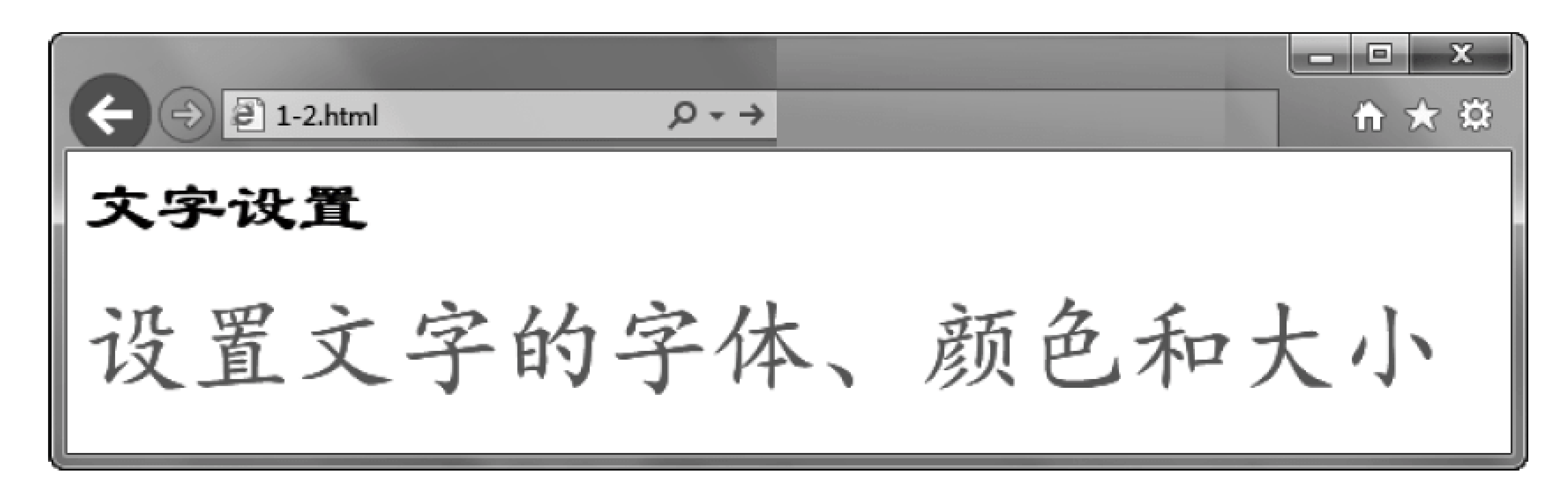


图 1-5　font 标记效果

3. 分段标记<p>

分段标记<p>...</p>定义了一个段落，使用该标记后续内容隔一行显示。若同时使用<p>和</p>，则将段落包围起来，表示一个分段的块；若省略结束标记</p>，可以将开始标记<p>放在段尾。

分段标记的常用属性是 align，用于设置段落的水平对齐方式。

示例代码如下。

```
<html>
<body>
<p align="left">这是段落一,左对齐.</p>
<p align="center">这是段落二,居中对齐.</p>
<p align="right">这是段落三,右对齐.</p>
</body>
</html>
```

设置效果如图 1-6 所示。



图 1-6　p 标记效果

4. 换行标记

换行标记
是空标签，它没有结束标签，因此
...</br>是错误的。
标记只是简单地开始新的一行，与上一行内容在逻辑上还属于一段。

示例代码如下。

```
<html><body>
<p>
«草»<br/>离离原上草,<br/>一岁一枯荣。<br/>野火烧不尽,<br/>春风吹又生。
</p><p>
«登鹳雀楼»<br/>白日依山尽,<br />黄河入海流。<br />欲穷千里目,<br />更上一层楼。
</p></body></html>
```

设置效果如图 1-7 所示。



图 1-7　br 标记效果

5. 有序列表

有序列表使用
...
和
...
标记来创建带序号的文本段落。<ol>标记有两个常用属性：start 和 type。start 属性用于数字序列的起始值，可以取整数值；type 属性用于设置规定在列表中使用的标记类型，其取值可以是 1、A、a、I、i。

示例代码如下。

```
<html><body>
<ol>
<li>足球</li>
<li>篮球</li>
<li>羽毛球</li>
</ol>
<ol start="50">
<li>游泳</li>
<li>滑冰</li>
<li>跳水</li>
</ol>
<ol type="A">
<li>跳高</li>
<li>跳远</li>
<li>跨栏</li>
</ol>
</body></html>
```

设置效果如图 1-8 所示。

6. 无序列表<ul>

无序列表使用
...
和
...
标记来创建无序号列表。<ul>标记的 type 属性用于指定列表项前面显示的项目符号，其取值可以是，disc：使用实心圆作为项目符号（默认值）；circle：使用空心圆作为项目符号；square：使用方块作为项目符号。

示例代码如下。

```
<html><body>
<h4>无序列表：</h4>
<ul>
<li>咖啡</li>
<li type="square">茶</li>
<li type="circle">牛奶</li>
</ul>
</body></html>
```

设置效果如图 1-9 所示。



图 1-8　ol 标记效果



图 1-9　ul 标记效果

7. 超链接标记<a>

超链接标记<a>...是用来定义超链接,可以实现从一个网页链接到另一个网页,或者链接到一个图像文件、一个需要下载的文件等。<a>标记最重要的属性是 href,它指示链接的目标。

在所有浏览器中,链接的默认外观是:未被访问的链接带有下划线且是蓝色的;已被访问的链接带有下划线且是紫色的;活动链接带有下划线且是红色的。



图 1-10　a 标记效果

示例代码如下。

```
<html><body>
<a href="http://www.sina.com.cn">新浪网</a>
<p>
<a href="..//images/1.jpge">图片浏览</a>
</body></html>
```

设置效果如图 1-10 所示。

8. 表格标记<table>

表格标记<table>...</table>用来定义 HTML 表格。简单的 HTML 表格由本标记以及一个或多个 tr、th 或 td 元素组成。tr 元素定义表格行,th 元素定义表头,td 元素定义表格单元。

示例代码如下。

```
<html><body>
<table border="1">
<tr>
<th>星期一</th>
<th>星期二</th>
<th>星期三</th>
<th>星期四</th>
<th>星期五</th>
</tr>
<tr>
<td>包子</td>
<td>馒头</td>
<td>花卷</td>
<td>油条</td>
<td>烧饼</td>
</tr>
<tr>
<td>大米粥</td>
<td>紫米粥</td>
<td>小米粥</td>
<td>豆浆</td>
<td>玉米粥</td>
</tr>
</table></body></html>
```

设置效果如图 1-11 所示。



星期一	星期二	星期三	星期四	星期五
包子	馒头	花卷	油条	烧饼
大米粥	紫米粥	小米粥	豆浆	玉米粥

图 1-11　table 标记效果

9. 表单标记<form>

表单是用来收集站点访问者信息的区域。表单可以包含允许用户进行交互的各种控件，如文本框、列表框、复选框和单选按钮等。通过输入文本、单击按钮与复选框、从下拉菜单中选择选项方式填写表单，站点访问者便送出所输入的数据，该数据就会根据所设置的表单处理程序，以各种不同的方式进行处理。

表单标记<form>...</form>用于为用户输入创建 HTML 表单。

常用的属性有以下几个。

① name：指定表单的名称。命名表单后，可以使用脚本语言（如 VBScript）来引用或控制该表单。

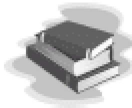
② method：指定将表单数据传输到服务器的方法，其取值可以是，post：在 HTTP 请求中嵌入表单数据；get：将表单数据附加到请求该页的 URL 中。

③ action：将要接收表单数据的动态网页的网址。

（1）在表单中添加单行文本框。

【基本语法】

```
< input type = "text" name = "字符串" value = "字符串" size = "n"  maxlength = "m">
```

【语法说明】

name 属性：文本框名称，可以在脚本中引用该文本。

value 属性：指定文本框的初始值。

size 属性：指定文本框的宽度。

maxlength：允许在文本框内输入的最大字符数。

（2）在表单中添加密码域。

【基本语法】

```
< input  type = "password"  name = "字符串"  value = "字符串"  size =  "n"  maxlength = "m">
```

【语法说明】

name 属性：用于指定密码域的名称。

value 属性：用于指定密码域的初始值。

size 属性：指定密码域的宽度。

maxlength 属性：用于指定允许在密码域内输入的最大字符数。

（3）在表单中添加复选框。

【基本语法】

```
< input type = "checkbox" name =  "字符串" value  = "字符串" [checked]>选项文本
```

【语法说明】

name：指定复选框对象的名称。

value：每一个复选框必须有且仅有一个 value，当被选择时这个值便会传送给表单处理程序。

checked：指定对应复选框为默认选中项。

(4) 在表单中添加单选按钮。

【基本语法】

```
< input type = "radio" name = "字符串" value = "字符串" [checked]>选项文本
```

【语法说明】

name 属性：指定单选按钮的名称，若干个名称相同的单选按钮构成一个控件组，在该组中只能选中一个选项。

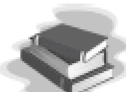
value 属性：一组相关的 radio 的 name 属性值要相同，但 value 值要不同，可让使用者任选其一。

checked 属性：是可选的，若使用该属性，则当第一次打开表单时该单选按钮处于选中状态。

(5) 在表单中生成一个命令按钮。

【基本语法】

```
< input type = "类型" name = "对象名" value = "字符串" [onclick = "子程序名"]>
```

【语法说明】

type：定义命令按钮的类型。当类型值为 submit 时，创建一个提交按钮；当值定义为 reset 时，创建一个重置按钮；当值定义为 button 时，创建一个自定义按钮。表单中添加自定义按钮后，应该为其编写脚本。

name：指定命令按钮的名称。

value：设定命令按钮上显示的文本内容。

onclick：指定单击命令按钮时系统自动调用并执行的函数或过程。

(6) 多行多列的文本框。

【基本语法】

```
< textarea   name = "对象名" [ cols = n] [rows = n] [readonly]>文本区中的字符串</textarea >
```

【语法说明】

name：指定多行文本框的名称。

cols：设定文字区块的字符宽度。

rows：设定文字区块的列数，即其高度。

readonly：设定多行文本框中的内容为只读。

(7) 列表框。

【基本语法】

```
<select   name="对象名"  >
<option   value=可选项 1 的值   [selected]>可选项 1 的提示</option>
<option   value=可选项 2 的值   [selected]>可选项 2 的提示</option>
⋮
</select>
```

【语法说明】

<select>标记的 name 属性用于指定表单元素的名称。

<option>标记用来在由<select>标记所指示的列表框中指示一个选项。

<option>标记属性的含义如下。

value：指定某一选项的值。可以自行修改，表单处理程序中接收的是此属性传送的值。但不同选项必须有不同的值。

selected：指定某选项为默认选中项。如果不指定此参数，则第一项为默认选项。

示例代码如下。

```
<html><body>
<form id="form1" name="form1" method="post" action="success.asp">
<table width="600" border="0" align="center" cellpadding="0" cellspacing="0">
<trbgcolor="#eeeeee">
<td height="20"><p class="STYLE3">当前位置: 用户注册</p></td>
</tr><tr>
<td height="30" colspan="2" bgcolor="#FFFFFF">姓名:
<input name="name" type="text" id="name" /></td>
</tr><tr>
<td height="30" colspan="2" bgcolor="#FFFFFF">性别:
<label><input type="radio" name="sex" value="男" />男</label>
<label><input type="radio" name="sex" value="女" />女</label>
</td></tr><tr>
<td height="30" colspan="2" bgcolor="#FFFFFF">出生日期:
<select name="year" size="1" id="year">
option value="1980">1980</option>
<option value="1981">1981</option>
⋮
<option value="1984">1985</option>
</select>年
<select name="month" size="1" id="month">
<option value="1" selected="selected">1</option>
<option value="2">2</option>
⋮
<option value="12">12</option>
</select>月
<select name="day" size="1" id="day">
```

```
<option selected="selected">1</option>
<option>2</option>
 ⋮
<option>31</option>
</select>日</td></tr><tr>
<td height="30" colspan="2" bgcolor="#FFFFFF">密码:
<input name="password" type="password" id="password" size="10" maxlength="10" />
 确认密码:
<input name="textfield" type="password" size="10" maxlength="10" />
(6—10 位)</td>
</tr><tr><tr>
<td height="30" colspan="2" bgcolor="#FFFFFF"><p>个人兴趣:<br />
<input type="checkbox" name="checkbox" value="读书" />读书
<input name="checkbox" type="checkbox" id="checkbox" value="旅游" />
旅游  
<input name="checkbox" type="checkbox" id="checkbox" value="音乐" />
音乐  
<input name="checkbox" type="checkbox" id="checkbox" value="绘画" />绘画<br /></tr>
<tr>
<td height="30" colspan="2" bgcolor="#FFFFFF">E-mail:
<input type="text" name="textfield2" />
</td></tr></table></Form>
<p>请单击确认按钮,输入会发送到服务器上名为 " success.asp " 的页面。</p>
</body></html>
```

设置效果如图 1-12 所示。

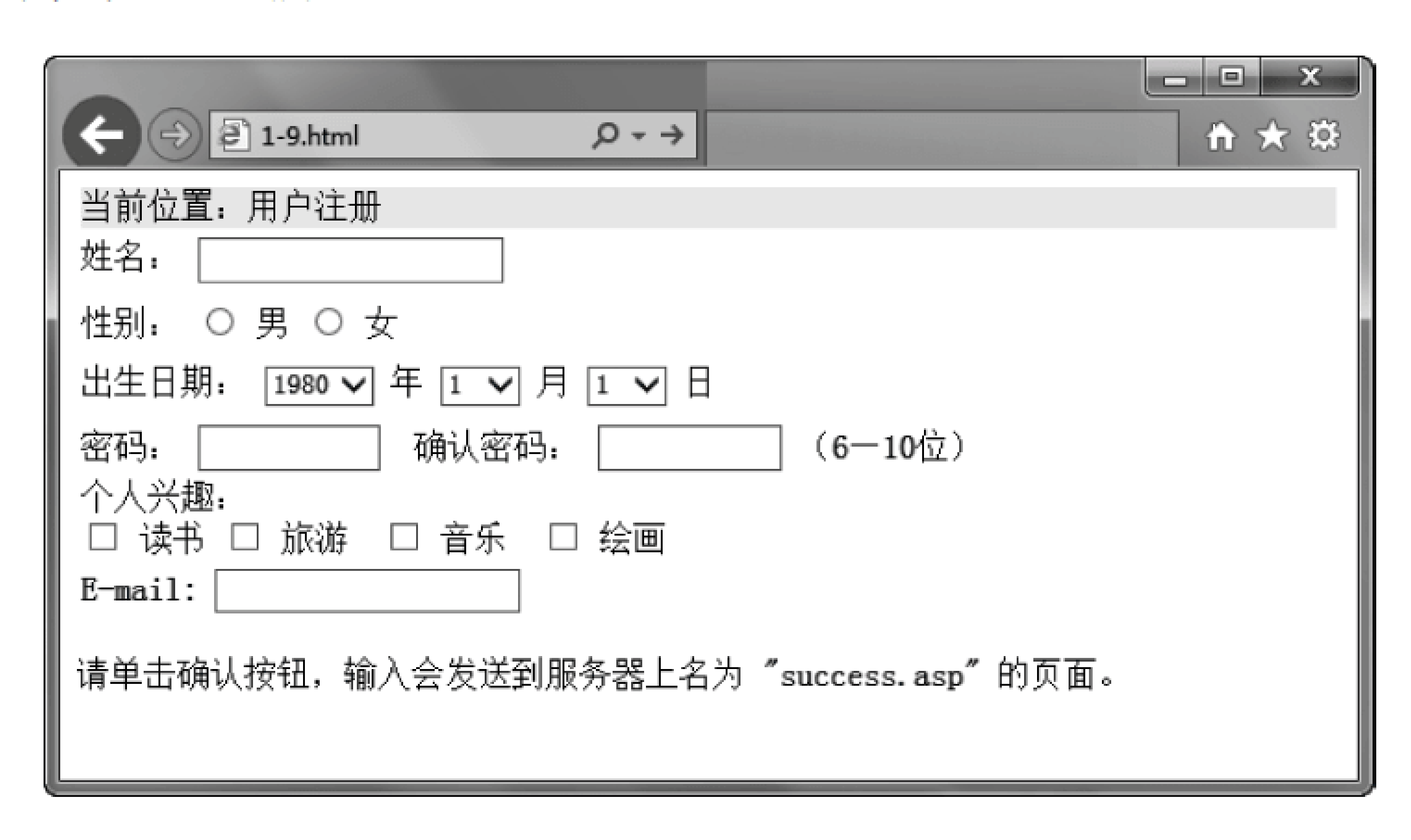


图 1-12 form 标记效果

1.3 动态网页

1.3.1 动态网页的概念

1. 动态网页概述

动态网页是指跟静态网页相对的一种网页编程技术。静态网页随着 HTML 代码的生成,页面的内容和显示效果就基本上不会发生变化了——除非修改页面代码。而动态

网页则不然，页面代码虽然没有变化，但是显示的内容却是可以随着时间、环境或者数据库操作的结果而发生改变的。

动态网页与网页上的各种动画、滚动字幕等视觉上的动态效果没有直接关系，动态网页可以是纯文字内容，也可以是包含各种动画的内容，这些只是网页具体内容的表现形式，无论网页是否具有动态效果，只要是采用了动态网站技术生成的网页都可以称为动态网页。

2. 动态网页的特点

(1) 动态网页一般以数据库技术为基础，可以大大降低网站维护的工作量。

(2) 动态网页具有交互性，网页能根据客户的要求和选择而动态地改变页面显示效果。

(3) 动态网页能够自动更新，无须动手更新 HTML 文档，就能自动生成新的页面。

(4) 动态网页实际上并不是独立存在于服务器上的网页文件，只有当用户请求时服务器才返回一个完整的网页。

(5) 动态网页的网站在进行搜索引擎推广时需要做一定的技术处理才能适应搜索引擎的要求；否则不易被检索。

(6) 动态网页在访问速度上不如静态网页，访问的人数过多，页面的加载速度就会变慢，对服务器来说也是一种负担。

(7) 动态网页以.aspx、.asp、.jsp、.php、.perl、.cgi 等常见形式为后缀。

1.3.2 动态网页开发语言

目前实现动态网页的技术主要有 ASP、ASP.NET、PHP 和 JSP。无论采用何种语言都能够实现动态网页的开发。

ASP 是目前最简单易学的动态网页技术，其功能足可以胜任绝大多数的应用需要，这也是它具有生命力的主要原因。

ASP.NET 是 ASP 的新一代产品，采用 C# 或 VB.NET 等新一代编程语言来实现流程控制。其功能强大，但学习难度比 ASP 要高一些。

PHP 采用 PHP 脚本语言来实现，该脚本语言是一种函数式的语言，与 C 语言很类似。PHP 支持跨平台运行。

JSP 采用 Java 语言来实现流程控制，功能强大，运行速度快，但与 ASP.NET 一样，学习难度较大。JSP 支持跨平台运行。

下面分别介绍一下这 4 种语言。

1. ASP 语言

ASP(Active Server Page，动态服务器页面)是微软公司开发的代替 CGI 脚本程序的一种应用，它可以与数据库和其他程序进行交互，是一种简单、方便的编程工具。ASP 网页文件的格式是.asp。

ASP 是 Microsoft 公司在 1996 年推出的一种运行于服务器端、嵌入了服务器端脚本的 Web 应用程序开发技术，内含于 IIS 3.0 以上的版本中。在 IIS 5.0 中支持 ASP 3.0，

同时也支持 ASP 2.0。

1) ASP 语言的优点

(1) 使用 VBScript、JScript 等简单易懂的脚本语言,结合 HTML 代码,即可快速地完成网站的应用程序。

(2) 无须编译,容易编写,可在服务器端直接执行。

(3) 与浏览器无关(Browser Independence),客户端只要使用可执行 HTML 码的浏览器,即可浏览用 ASP 所设计的网页内容。

(4) 能与任何 ActiveX Scripting 语言兼容。除了可使用 VBScript 或 JScript 语言来设计外,还可通过 plug-in 的方式,使用由第三方所提供的其他脚本语言,如 REXX、Perl、Tcl 等。

(5) 可使用服务器端的脚本来产生客户端的脚本。

(6) ActiveX Server Components(ActiveX 服务器组件)具有无限可扩充性。

2) ASP 语言的缺点

(1) Windows 本身的所有问题都会一成不变地累加到它的身上。安全性、稳定性、跨平台性都会因为与 Windows 的捆绑而显现出来。

(2) 由于 ASP 使用了 COM 组件,所以它会变得十分强大,但这样的组件或是在操作中一不注意,外部攻击就可以取得相当高的权限而导致网站瘫痪或者数据丢失。

(3) 由于 ASP 还是一种 Script 语言,所以除了大量使用组件外,没有其他办法提高其工作效率。

(4) 无法实现跨操作系统的应用。

(5) 无法完全实现一些企业级的功能,如完全的集群、负载均衡。

2. JSP 语言

JSP(Java Server Pages,Java 服务器页面)是由 Sun Microsystems 公司倡导、许多公司参与,一起建立的一种动态网页技术标准。

JSP 技术有点类似 ASP 技术,它是在传统的网页 HTML 文件中插入 Java 程序段和 JSP 标记,从而形成 JSP 文件,后缀名为 *.jsp。

用 JSP 开发的 Web 应用是跨平台的,既能在 Linux 下运行,也能在其他操作系统上运行。它实现了 HTML 语法中的 Java 扩张。

JSP 技术使用 Java 编程语言编写类 XML 的 tags 和 scriptlets,来封装产生动态网页的处理逻辑。网页还能通过 tags 和 scriptlets 访问存在于服务端资源的应用逻辑。JSP 将网页逻辑与网页设计的显示分离,支持可重用的基于组件的设计,使基于 Web 的应用程序的开发变得迅速和容易。

JSP 是一种动态页面技术,它的主要目的是将表示逻辑从 Servlet 中分离出来。Java Servlet 是 JSP 的技术基础,而且大型的 Web 应用程序的开发需要 Java Servlet 和 JSP 配合才能完成。JSP 具备了 Java 技术的简单易用,完全面向对象,具有平台无关性且安全可靠,主要面向因特网的所有特点。

1) JSP 语言的优点

(1) 一次编写,随处运行。

(2) 系统的多平台支持。基本上可以在所有平台上的任意环境中开发,在任意环境中进行系统部署和扩展。

(3) 强大的可伸缩性。从只有一个小的 Jar 文件就可以运行 Servlet/JSP,到由多台服务器进行集群和负载均衡,到多台 Application 进行事务处理、消息处理,一台服务器到无数台服务器,Java 显示了一个巨大的生命力。

(4) 多样化和功能强大的开发工具支持。

(5) 支持服务器端组件。Web 应用需要强大的服务器端组件来支持,开发人员需要利用其他工具设计实现复杂功能的组件供 Web 页面调用,以增强系统性能。JSP 可以使用成熟的 Java BEANS 组件来实现复杂的商务功能。

2) JSP 语言的缺点

(1) 与 ASP 一样,Java 的一些优势正是它致命的问题所在。正是由于为了跨平台的功能,为了极度的伸缩能力,所以极大地增加了产品的复杂性。

(2) Java 的运行速度是用 class 常驻内存来完成的,所以它在一些情况下所使用的内存比较大。

3. PHP 语言

PHP(HyperText Preprocessor,超文本预处理器)是一种通用开源脚本语言。PHP 于 1994 年由 RasmusLerdorf 创建,刚开始是 RasmusLerdorf 为了要维护个人网页而制作的一个简单的用 Perl 语言编写的程序。这些工具程序用来显示 RasmusLerdorf 的个人履历以及统计网页流量。后来又用 C 语言重新编写,包括可以访问数据库。它将这些程序和一些表单直译器整合起来,称为 PHP/FI。PHP/FI 可以和数据库连接,产生简单的动态网页程序。

PHP 语法吸收了 C 语言、Java 和 Perl 的特点,利于学习,使用广泛,主要适用于 Web 开发领域。PHP 独特的语法混合了 C、Java、Perl 及 PHP 自创的语法。它可以比 CGI 或者 Perl 更快速地执行动态网页。

用 PHP 做出的动态页面与其他的编程语言相比,PHP 是将程序嵌入 HTML(标准通用标记语言下的一个应用)文档中去执行,执行效率比完全生成 HTML 标记的 CGI 要高许多;PHP 还可以执行编译后代码,编译可以达到加密和优化代码运行,使代码运行更快。

1) PHP 语言的优点

(1) 跨平台,性能优越,可以和很多免费的平台结合。

(2) 语法简单,如果学习过 C 和 Perl 则很容易上手,并且跟 ASP 有部分类似。

(3) 有比较完整的支持,比如使用 ADODB 或者 PEAR::DB 做数据库抽象层,用 Smarty 或者 Smart Template 做模板层,如果是 PHP 5.1 的话,还能够使用 PDO(PHP Data Object)来访问数据库。

(4) 有很多成熟的框架,如支持 MVC 的框架 phpMVC、支持类似 ASP.NET 的事件驱动的框架 Prado、支持类似 Ruby On Rails 的快速开发的框架 Cake 等,足够满足用户的应用需求。

(5) 有很多开源的框架或开源的系统可以使用,如比较知名的开源框架有 Zend

Framework、CakePHP、CodeIgniter、symfony 等。

2）PHP 语言的缺点

（1）对多线程支持不太好，大多数时候只能简单地模拟去实现。

（2）语法不太严谨，如变量不需要定义就可以使用，在 C、Java、C++ 中变量是必须先定义以后才可以使用的。

4．ASP.NET 语言

ASP.NET 是微软公司.NET Framework 的一部分，它是一种使嵌入网页中的脚本可由因特网服务器执行的服务器端脚本技术，可以在通过 HTTP 请求文档时再在 Web 服务器上动态创建它们。

ASP.NET 的前身 ASP 技术，是在 IIS 2.0 上首次推出，当时与 ADO 1.0 一起推出，在 IIS 3.0 发扬光大，成为服务器端应用程序的热门开发工具，微软还特别为它量身打造了 VisualInter Dev 开发工具。

目前很多人对 ASP.NET 和 ASP 概念混淆，其实两者是不同的。ASP 是解释型编程框架，而 ASP.NET 是编译型框架；ASP.NET 无论是从执行效率和安全上都远远超过 ASP；ASP 文件的后缀是.asp，而 ASP.NET 则是.aspx 和.aspx.cs。ASP.NET 实现了代码分离，让代码管理更加直观。具体区别如下。

1）开发语言不同

ASP 仅局限于使用 non-type 脚本语言来开发，用户给 Web 页中添加 ASP 代码的方法与客户端脚本中添加代码的方法相同，导致代码杂乱。

ASP.NET 允许用户选择并使用功能完善的 strongly-type 编程语言，也允许使用潜力巨大的.NET Framework。

2）运行机制不同

ASP 是解释运行的编程框架，所以执行效率较低。

ASP.NET 是编译型的编程框架，运行的是服务器上编译好的公共语言运行库代码，可以利用早期绑定、实施编译来提高效率。

3）开发方式

ASP 把界面设计和程序设计混在一起，维护和重用困难。

ASP.NET 把界面设计和程序设计以不同的文件分离开，复用性和维护性得到了提高。

ASP.NET 语言的优点如下。

1）增强的性能

ASP.NET 是在服务器上运行的编译好的公共语言运行库代码。与被解释的 ASP 不同，ASP.NET 可利用早期绑定、实时编译，优化本机服务。这相当于在编写代码行之前便显著地提高了性能。

2）世界级的工具支持

ASP.NET Framework 补充了 Visual Studio 集成开发环境中的大量工具箱和设计器。WYSIWYG 编辑、拖放服务器控件和自动部署只是这个强大工具所提供功能中的少数几种。

3）威力和灵活性

由于 ASP.NET 基于公共语言运行库，因此 Web 应用程序开发人员可以利用整个平台的威力和灵活性。.NET Framework 类库、消息处理和数据访问解决方案都可从 Web 无缝访问。ASP.NET 也与语言无关，所以可以选择最适合应用程序的语言，或跨多种语言分割应用程序。

另外，公共语言运行库的交互性保证在迁移到 ASP.NET 时保留基于 COM 的开发中的现有投资。

4）简易性

ASP.NET 使执行常见任务变得容易，从简单的窗体提交和客户端身份验证到部署和站点配置。例如，ASP.NET 页框架使用户可以生成将应用程序逻辑与表示代码清楚分开的用户界面，和在类似 Visual Basic 的简单窗体处理模型中处理事件。

另外，公共语言运行库利用托管代码服务（如自动引用计数和垃圾回收）简化了开发。

5）可管理性

ASP.NET 采用基于文本的分层配置系统，简化了将设置应用于服务器环境和 Web 应用程序。由于配置信息是以纯文本形式存储的，因此可以在没有本地管理工具帮助的情况下应用新设置。

此“零本地管理”哲学也扩展到了 ASP.NET Framework 应用程序的部署。只需将必要的文件复制到服务器，即可将 ASP.NET Framework 应用程序部署到服务器。不需要重新启动服务器，即使是在部署或替换运行的编译代码时。

6）可缩放性和可用性

ASP.NET 在设计时考虑了可缩放性，增加了专门用于在聚集环境和多处理器环境中提高性能的功能。另外，进程受到 ASP.NET 运行库的密切监视和管理，以便当进程行为不正常（泄露、死锁）时，可就地创建新进程，以帮助保持应用程序始终可用于处理请求。

7）自定义性和可扩展性

ASP.NET 随附了一个设计周到的结构，它使开发人员可以在适当的级别“插入”代码。实际上，可以用自己编写的自定义组件扩展或替换 ASP.NET 运行库的任何子组件。实现自定义身份验证或状态服务一直没有变得更容易。

8）安全性

借助内置的 Windows 身份验证和基于每个应用程序的配置，可以保证应用程序的安全。

ASP.NET 语言的缺点如下。

（1）数据库的连接复杂。

（2）不具有跨平台性，只支持 Windows 平台。

（3）在内存使用和执行时间方面耗费非常大。

本书采用 ASP.NET 作为动态网页编程语言，后续的环境配置和网页编写都以 ASP.NET 为例。

1.4 网站运行环境

1.4.1 硬件和操作系统要求

ASP.NET 运行环境要求如下。

硬件要求：CPU 推荐使用 1GHz 以上，内存推荐 2GB 以上，显示器推荐 1024×768 像素，增强色为 16 位。

操作系统要求：应为 Windows Server 2003 Service Pack 2、Windows Server 2008、Windows 7 或 Windows 8 等(注：支持 ASP.NET 的 IIS 服务配置)。

1.4.2 Windows 7 下配置 ASP.NET 服务器 IIS

1. 安装 IIS

IIS(Internet Information Services，互联网信息服务)是由微软公司提供的基于运行 Microsoft Windows 的互联网基本服务。最初是 Windows NT 版本的可选包，随后内置在 Windows 2000、Windows XP Professional 和 Windows Server 2003 一起发行，但在 Windows 2007 版本上并没有 IIS，需要自己安装 IIS，过程如下。

(1) 选择“开始”→“控制面板”→“程序和功能”命令，打开“程序和功能”窗口，在该窗口中单击“打开或关闭 Windows 功能”，如图 1-13 所示。



图 1-13 “程序和功能”窗口

(2) 弹出图 1-14 所示对话框。找到“Internet 信息服务”，选中“Web 管理工具”下的所有复选框。

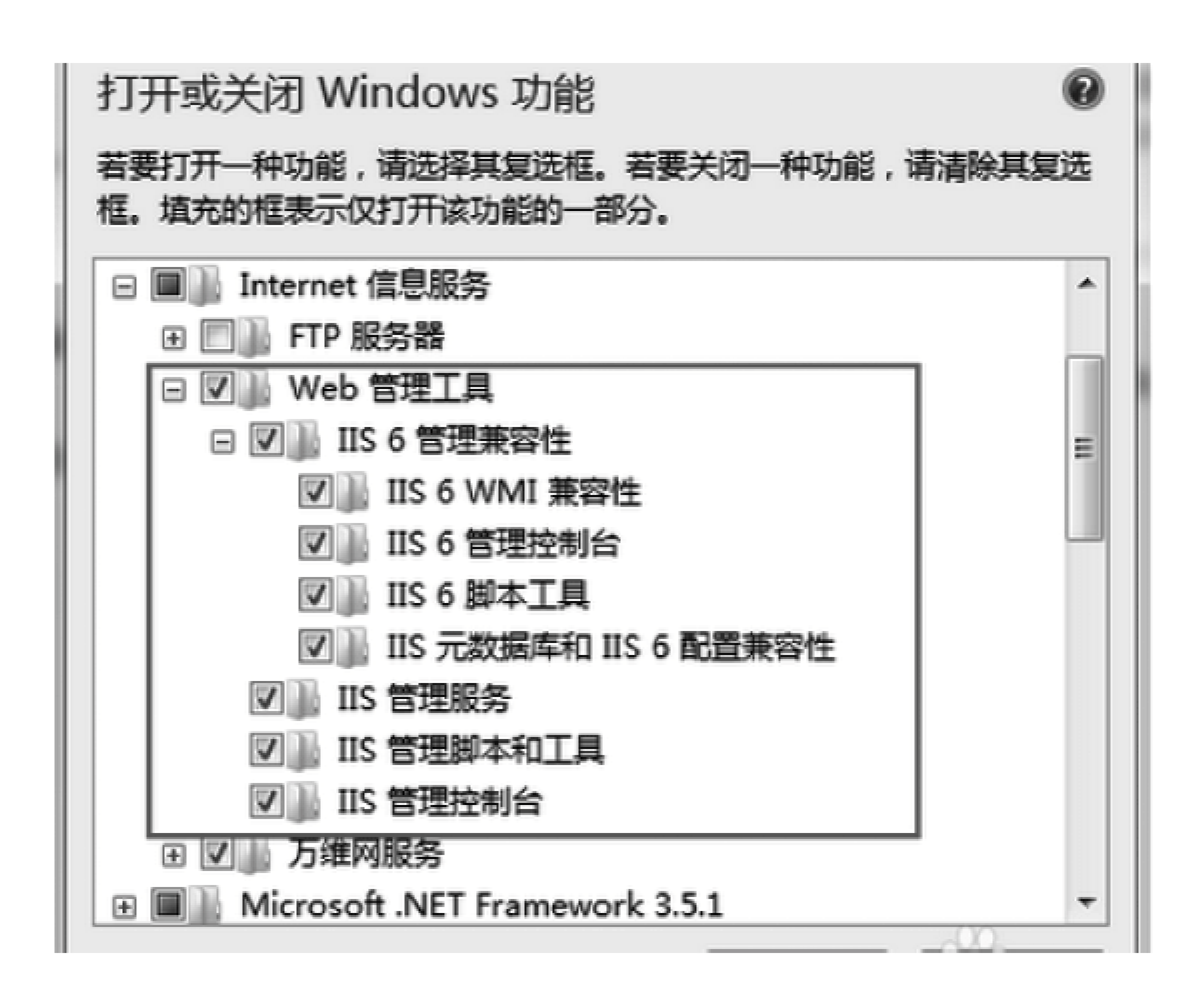


图 1-14　选中“Web 管理工具”下的所有复选框

(3) 在“打开或关闭 Windows 功能”对话框中同时选中“万维网服务”下的“应用程序开发功能”下的 ASP.NET 复选框，如图 1-15 所示，其他保持默认即可，然后单击“确定”按钮开始安装。

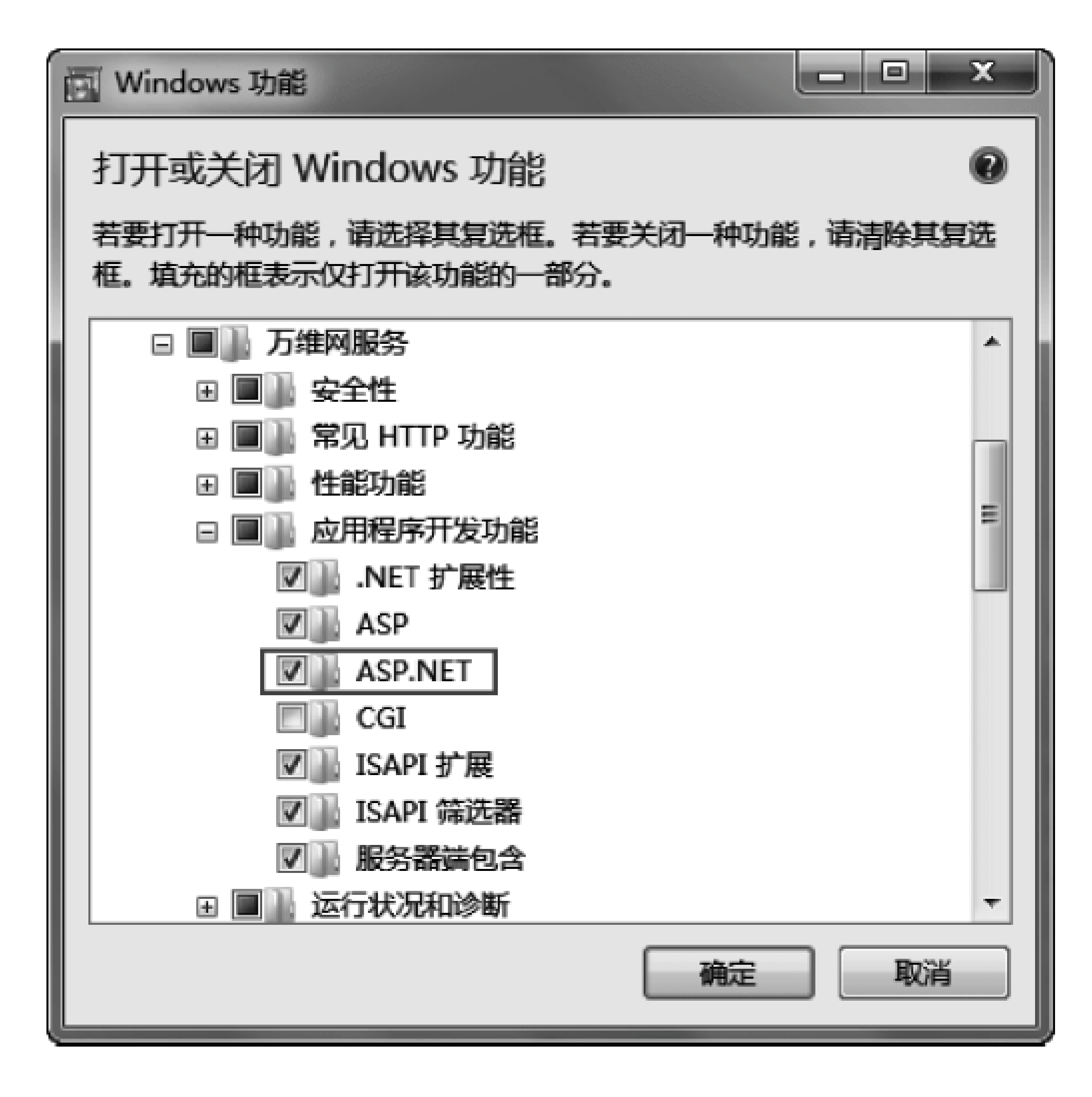


图 1-15　选中 ASP.NET 复选框

2. 配置管理

(1) 在“控制面板”中，单击“管理工具”，打开“管理工具”窗口，在该窗口中双击“Internet 信息服务(IIS)管理器”选项，如图 1-16 所示。

(2) 双击“Internet 信息服务(IIS)管理器”选项打开管理器窗口，即进入 ASP.NET 相关配置，如图 1-17 所示。

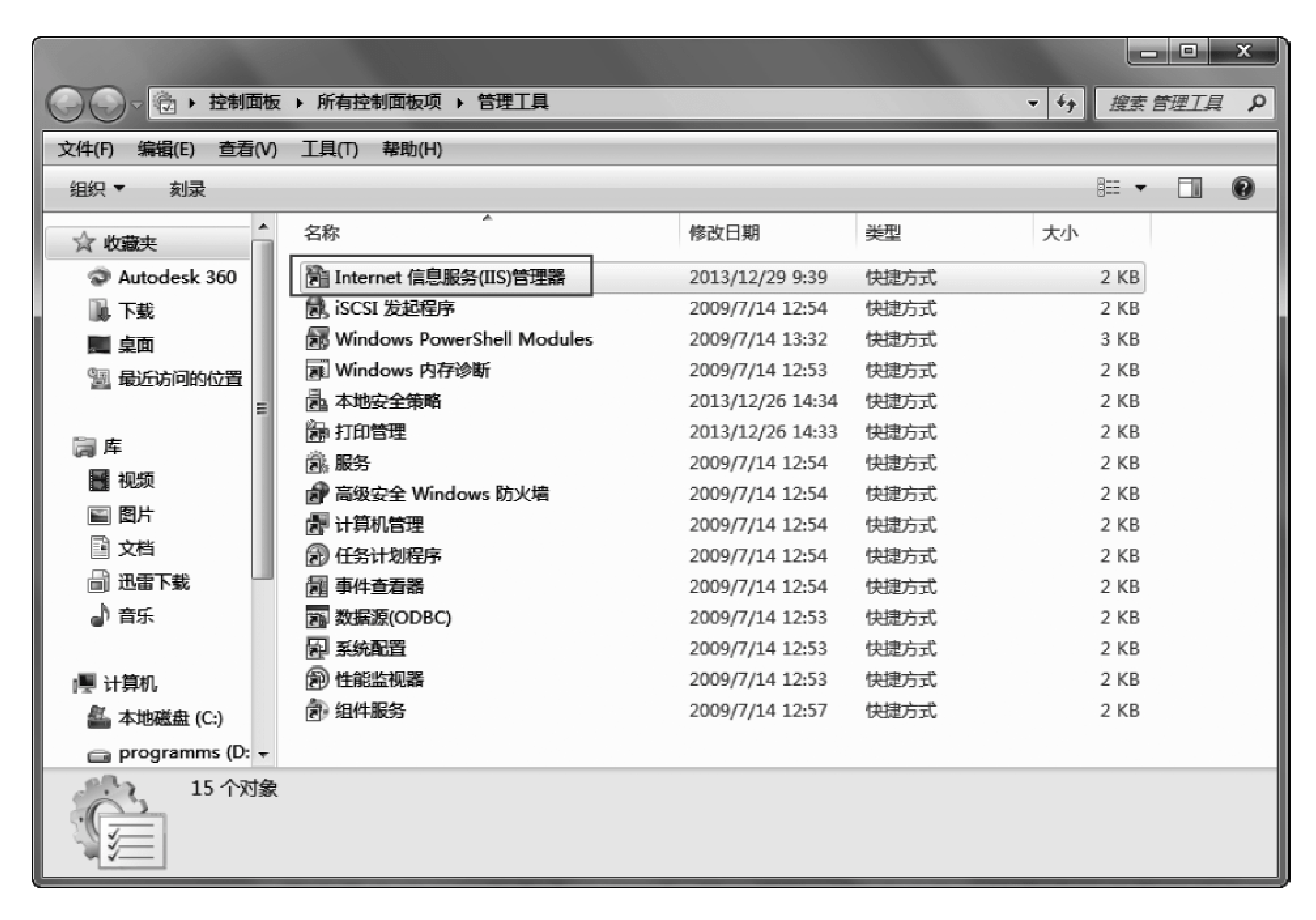


图 1-16 “管理工具”窗口

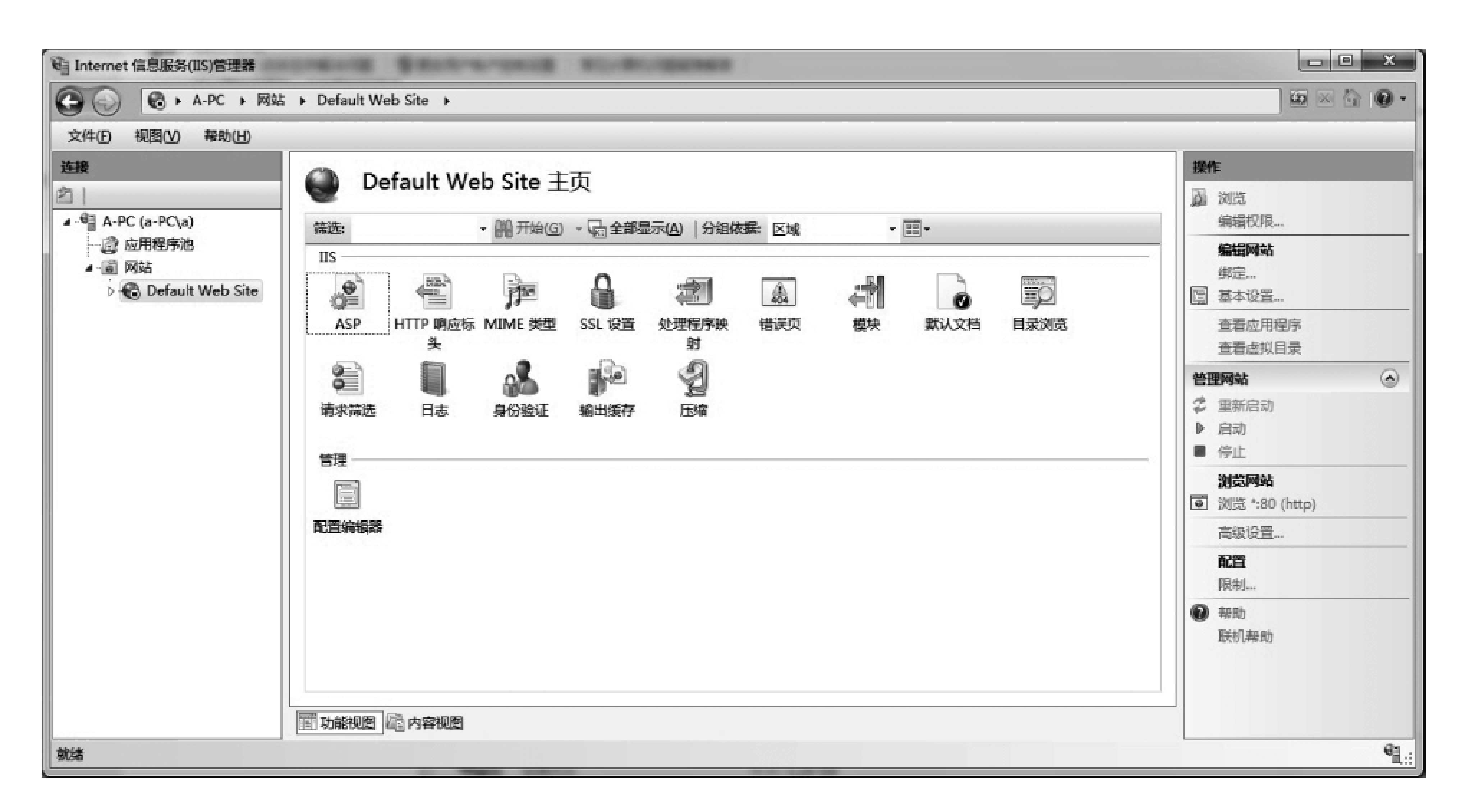
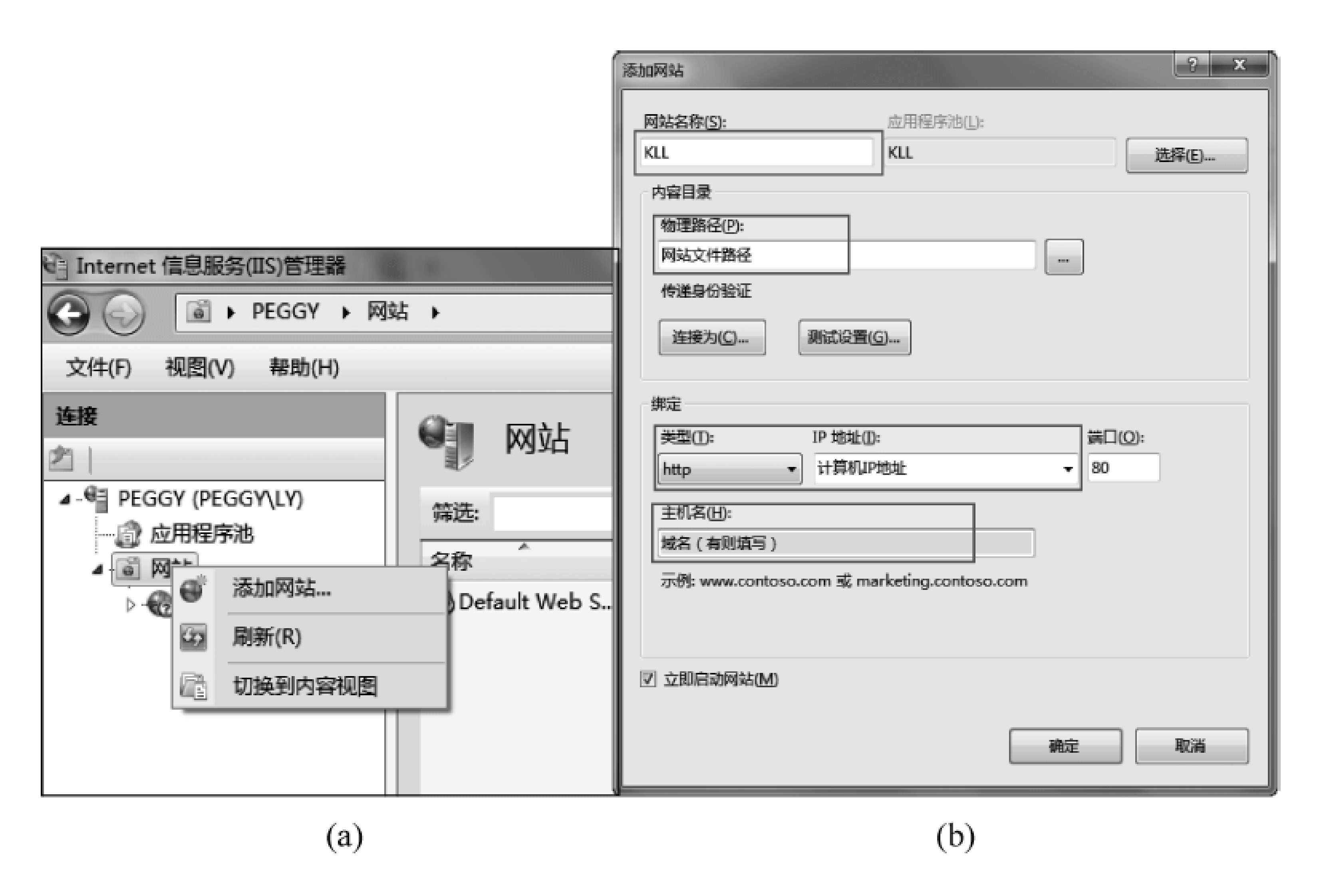


图 1-17 Internet 信息服务(IIS)管理器

(3) 创建网站。右击 IIS 管理器左侧窗格的“网站”,在快捷菜单中选择“添加网站”命令,如图 1-18(a)所示。打开“添加网站”对话框,如图 1-18(b)所示。

(4) 在“添加网站”对话框中,设置相关参数。例如,网站名称为 MyFirstWeb,物理路径为 F:\website\MyFirstWeb,IP 地址为 192.168.1.100。单击“确定”按钮,配置好 IIS 环境。ASP.NET 的环境配置将在后续章节讲解。



(a)　　　　　　　　(b)

图 1-18　添加网站



本章小结

本章重点介绍了超文本语言 HTML 的各种标记的含义和使用方法，以及常用动态网页开发语言及其优缺点。通过本章的学习，同学们应该能够利用 HTML 编写简单的网页，并通过浏览器进行浏览，学会配置网站运行环境。其中重点掌握表格、超链接、图形、表单等标记的使用和标记属性的设置，为后续的课程打下坚实的基础。



思考与练习

1. 选择题

(1) HTML 指的是(　　)。

A. 超文本标记语言(HyperText Markup Language)

B. 家庭工具标记语言(Home Tool Markup Language)

C. 超链接和文本标记语言(Hyperlinks and Text Markup Language)

(2) 用 HTML 标记语言编写一个简单的网页，网页最基本的结构是(　　)。

A. <html><head>…</head><frame>…</frame></html>

B. <html><title>…</title><body>…</body></html>

C. <html><title>…</title><frame>…</frame></html>

D. <html><head>…</head><body>…</body></html>

(3) 以下标记符中，用于设置页面标题的是(　　)。

A. <title>　　　　B. <caption>

C. <head>　　　　D. <html>

(4) 以下标记符中，没有对应的结束标记的是(　　)。
A. <body>　　B.
　　C. <html>　　D. <title>

(5) 下面(　　)是换行符标记。
A. <body>　　B. <font>　　C.
　　D. <p>

2. 问答题

(1) 简述一个 HTML 文档的基本结构。

(2) 写出 URL 包含的 3 个部分内容的作用。

(3) 动态网页的编程语言都有什么？它们的优点和缺点有哪些？

第 2 章

C#

C#是一种新的、面向对象的编程语言,是.NET 框架中新一代的开发工具,用 C#编写的应用程序可以充分利用.NET 的框架体系带来的优点,既可以用来编写基于通用网络协议的 Internet 服务软件,也可以编写各种数据库、网络服务应用程序和 Windows 窗口界面程序。

它简化了 C++语言在类、命名空间、方法重载和异常处理等方面的操作,使用组件辅助编程,十分容易掌握。C#语法与 C++和 Java 语法非常相似,如果使用过 C++和 Java,学习 C#语言应是比较轻松的。本章对 C#语言的知识进行详细的介绍,旨在使读者对 C#语言有一个全面的了解。

2.1 .NET 开发环境

Microsoft.NET(以下简称.NET)框架是微软提出的新一代 Web 软件开发模型,.NET 开发平台包括编程语言(C#、Visual Basic、Visual C++等)、.NET 开发工具(Visual Studio.NET)以及.NET 框架(.NET Framework)。.NET 的最终目标就是让用户在任何地方、任何时间以及利用任何设备都能访问所需的信息、文件和程序。

.NET 已包含非常大的代码库,可以在通过面向对象语言,如 C#等编程技术中使用这些代码,这样可以极大地提高程序设计人员的工作效率。

2.1.1 Visual Studio 2013 的安装与注册

Visual Studio 是一个由基于组件的软件开发工具,内置了多种提高工作效率的功能,如自动补全方括号、使用快捷键移动整行或整块的代码以及行内导航。此外,Visual Studio 2013 的团队资源管理器增强了主页设计,可以更简便地导航到团队协作功能,并可取消停靠"挂起更改"和"生成",使其显示在一个单独的窗口中。

下面介绍 Visual Studio 2013 的安装过程。新旧版本 Visual Studio 是可以共存的,但是在安装过程中,旧版本的 Visual Studio 一定要先关闭。

(1) 双击图标 运行安装文件,显示如图 2-1 所示提示信息。

(2) 自定义选择安装路径时,注意所属路径的预留空间要充足;否则安装会失败。同意许可条款,进入下一步。

(3) 选择功能,如图 2-2 所示,在选择安装的可选功能里,可以根据自己需要选中,也可以默认全选。要注意预留空间,保证程序正常运行,然后单击“下一步”按钮开始安装。等待大概 30 分钟,就可以完成安装。安装过程中,Visual Studio 会占用很多的系统资源,所以最好不要开启其他软件,等待安装完成。出现“安装成功”提示信息后,重启计算机就可以使用了。



图 2-1 安装提示



图 2-2 选择功能

(4) 基本配置。第一次打开 Visual Studio 2013,需要进行一些基本配置,如开发设置、颜色主题,根据自己的需求设置,然后等待几分钟就可以使用。由于 Visual Studio 2013 引入了一种联网 IDE 体验,大家可以使用微软的账户登录,而且其还自动采用联网 IDE 体验的设备上同步设置,包括快捷键、Visual Studio 外观(主题、字体等)各种类别同步的设置,如图 2-3 所示。

(5) 注册产品。软件默认有效期为 30 天,需要完成注册过程才能长期使用。打开 Visual Studio 2013,在工具栏中找到“帮助”菜单,如图 2-4 所示,单击“注册产品”命令,会弹出一个对话框,里面会显示软件的注册状态,如图 2-5 所示,单击“更改我的产品许可证”,会弹出一个对话框,从中输入产品密钥,即可完成注册过程。

注册成功后,所有的操作基本完成,就可以正常使用了。

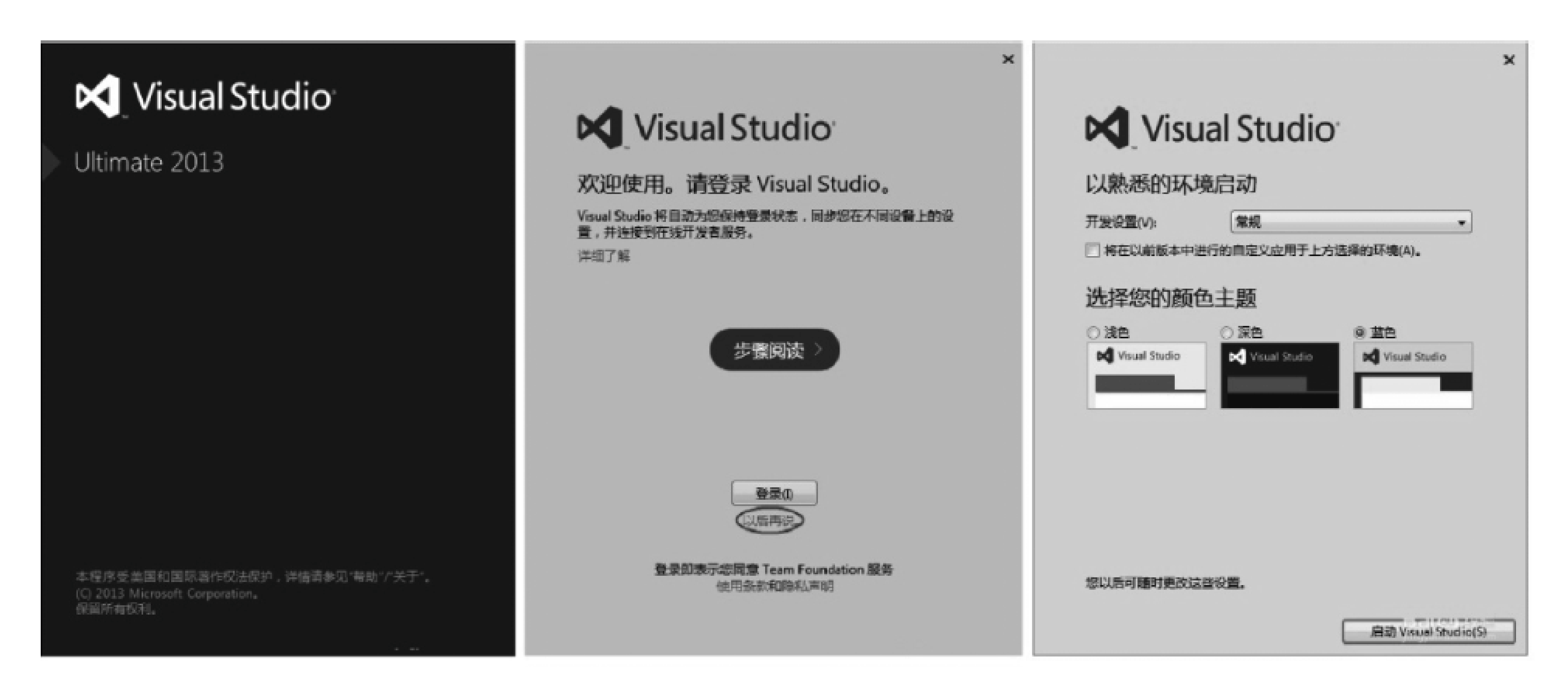


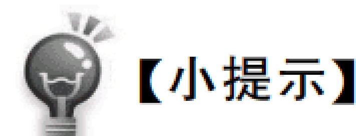
图 2-3　基本配置



图 2-4　显示状态



图 2-5　完成注册


【小提示】

安装过程注意事项：系统盘和安装路径的预留空间要充足；否则安装会失败。由于密钥过期，软件注册可能会失效，需要再次注册。新版本 VS 可能会因计算机配置的问题出现卡顿的现象，应尽量减少其他软件的使用。

2.1.2 .NET Framework 简介

Microsoft.NET Framework 是用于 Windows 的新托管代码编程模型。在今天的软件环境中，应用程序的来源很多，它们需要执行很多任务。对应用程序代码的信任是一个主要需求，因为谁也不想软件或信息遭到破坏。给予许可的安全策略，不允许对敏感信息的不适当访问，或将本地机器暴露给恶意的程序，甚至是有平常错误的代码。

在以往，安全结构提供了基于用户账号的隔离和访问控制方式，在这些限制内给予代码完全访问权，并假定由特定用户可运行的代码具有相同的信任度。但是，如果所有程序都代表某用户运行，根据用户对代码的隔离，对于保护一个程序不被其他用户使用是不够的。

另一种情况，不能被完全信任的代码经常被转移到“沙箱”模型中执行，在此，代码运行于隔离环境，而不会访问大部分的服务。在今天，应用程序成功的安全解决方案必须能强化两个安全模型间的平衡。它必须提供对资源的访问，以便完成有用的工作；它需要对应用程序的安全性做细致的控制以确保代码被识别、检测，并给予合适的安全级别。.NET Framework 就提供了一个这样的安全模型。

.NET Framework 安全解决方案基于管理代码的概念，以及由通用语言运行时(CLR)加强的安全规则。大部分管理代码需要进行验证以确保类型安全及预先定义好的其他属性的行为的安全。.NET Framework 的目的就是要让建立 Web Services 以及因特网应用程序的工作变得简单。

.NET Framework 包括了三大部分：第一个部分是 Common Language Runtime (CLR，所有.NET 程序语言公用的执行时期组件)；第二部分是共享对象类别库(提供所有.NET 程序语言所需要的基本对象)；第三部分是重新以组件的方式写成的(旧版本则是以 asp.dll 提供 ASP 网页所需要的对象)。

如图 2-6 所示，在.NET 体系结构中，.NET Framework 包含一个非常大的代码库，可以在客户语言(如 C#)中通过面向对象的编程技术来使用这些代码。这个库分为不同的模块，可以根据需要来使用其中的各个部分，如一个模块包含 Windows 应用程序的构件、一个模块包含 Web 开发的代码块等。

2.1.3 配置 ASP.NET 环境

在“网站”菜单上选择“ASP.NET 配置”。选择“安全”选项卡，单击“使用安全设置向导按部就班地配置安全性”链接，然后单击“下一步”按钮。

向导会显示一个页面，可以在其中选择网站将使用的身份验证方法。选择“通过 Internet”选项。此选项指定用户的网站将使用依赖于 ASP.NET 成员资格系统的

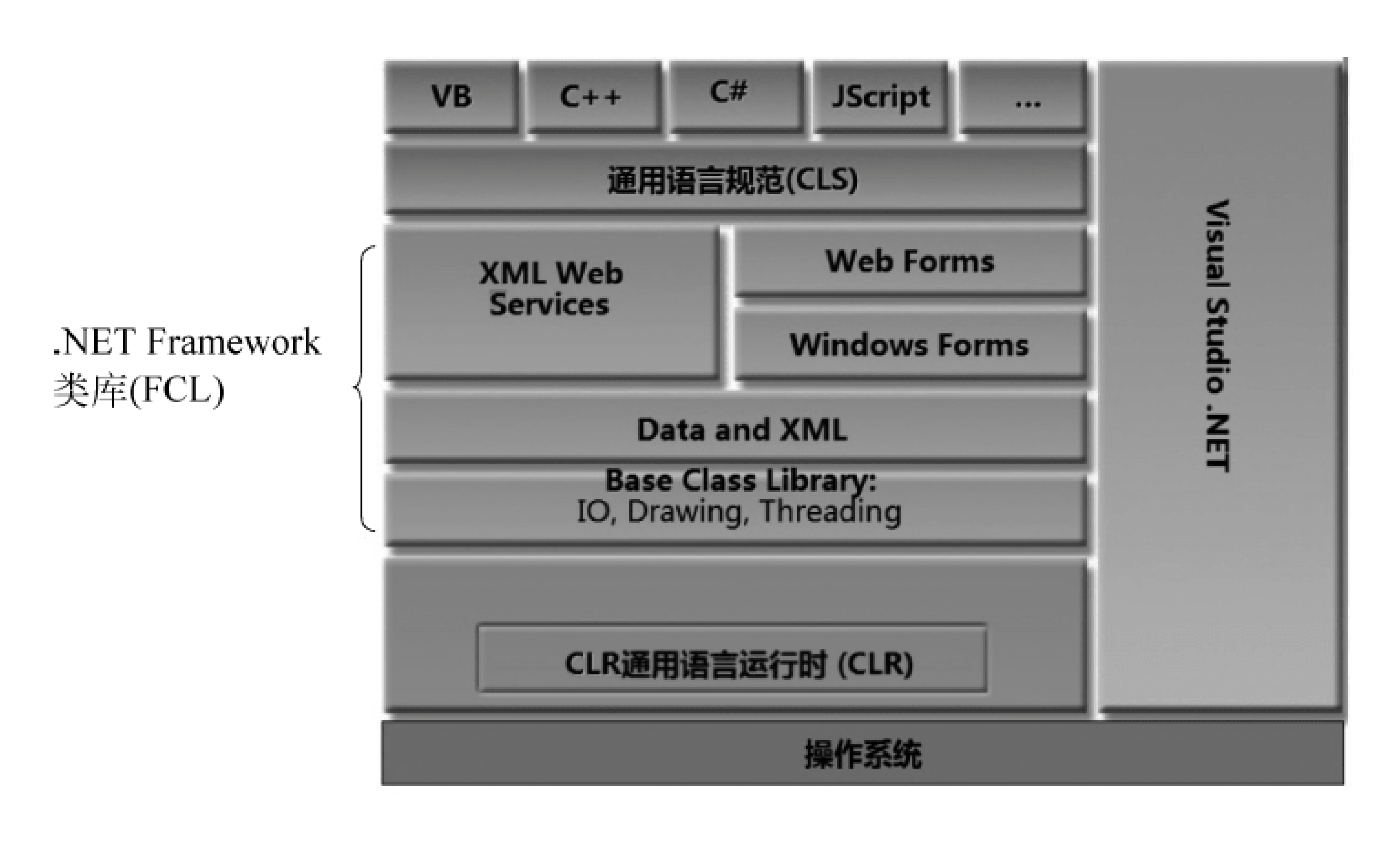


图 2-6 .NET 的体系结构

ASP.NET Forms 身份验证。当使用 Forms 身份验证时,用户将使用在本演练前面部分中创建的登录页登录到网站。



【小提示】

“通过局域网”选项可将网站配置为使用 Windows 身份验证,这在只有企业网络中的人员才能访问网站的情况下很实用。这是 ASP.NET 成员资格的默认设置。

单击“下一步”按钮。该向导指示将应用程序配置为使用高级提供程序设置。默认情况下,成员资格信息存储在网站上 App_Data 文件夹的 Microsoft SQL Server 数据库文件中。单击“下一步”按钮,取消选中“为此网站启用角色”复选框,再单击“下一步”按钮。在这个显示页中,可以创建新用户。输入定义网站用户的信息。相关属性含义如下。

(1) User Name:名称(不能有空格)。

(2) Password:密码。需要严格的密码(该密码包括大写和小写字母以及标点,且长度至少为 8 个字符)。

(3) E-mail:个人电子邮件地址。如果必须重置之前设置密码,则此电子邮件地址将用于发送新的密码。因此,需要输入一个有效的电子邮件地址。

(4) Security Question 和 Security Answer:输入可在以后需要重置密码时使用的问题和答案。

最后,选中“活动用户”复选框。单击“创建用户”按钮,即可完成设置。

2.2 面向对象

面向对象的思想已经涉及软件开发的各个方面。如面向对象的分析(Object Oriented Analysis,OOA)、面向对象的设计(Object Oriented Design,OOD)以及经常说的面向对象的编程实现(Object Oriented Programming,OOP)。本章将学习面向对象的知识。

2.2.1 概念

面向对象程序设计(OOP)是一种程序设计范型,同时也是一种程序开发的方法。已经被证实的是,面向对象程序设计推广了程序的灵活性和可维护性,并且在大型项目设计中广为应用。此外,支持者声称面向对象程序设计要比以往的做法更加便于学习,因为它能够让人们更简单地设计并维护程序,使得程序更加便于分析、设计、理解。

当提到面向对象时,它不仅指一种程序设计方法,更多意义上是一种程序开发方式。在这方面,必须了解更多关于面向对象系统分析和面向对象设计方面的知识。

面向对象的设计是一种提供符号设计系统的面向对象的实现过程,它用非常接近实际领域术语的方法把系统构造成"现实世界"的对象,是把分析阶段得到的需求转变成符合成本和质量要求的、抽象的系统实现方案的过程。

面向对象程序设计可以看作是一种在程序中包含各种独立而又互相调用的对象的思想,这与传统思想刚好相反:传统程序设计主张将程序看作一系列函数的集合,或者直接就是一系列对计算机下达的指令。而面向对象程序设计中的每一个对象都能够接收数据、处理数据并将数据传达给其他对象,因此它们都可以被看作一个小型的"机器",即对象。这也是面向对象的主要思想。

1. 对象

对象的概念是面向对象技术的核心。对象是人们要进行研究的任何事物,从最简单的整数到复杂的飞机等均可看作对象,它不仅能表示具体的事物,还能表示抽象的规则、计划或事件。不管是有生命的对象还是无生命的对象,它们都有一些共同的特征,即都有属性和行为。

软件对象是仿照现实对象建立的,它们也有状态和行为。软件对象用变量表示对象的状态,用方法实现对象的行为。一个对象就是变量和相关方法的集合。

2. 封装

面向对象程序设计方法提出了一个全新的概念——类,它的主要思想是将数据(数据成员)及处理这些数据的相应方法(函数成员)封装到类中,类的实例则称为对象。这就是常说的封装性。封装性是保证软件部件具有优良模块性的基础。面向对象的类是封装良好的模块,类定义将其说明(用户可见的外部接口)与实现(用户不可见的内部实现)显式地分开,其内部实现按其具体定义的作用域提供保护。

对象是封装的最基本单位。封装防止了程序相互依赖性而带来的变动影响。面向对象的封装比传统语言的封装更清晰、更有力。

一个对象的变量构成这个对象的核心,一般不将其对外公开,而是将对变量的处理方法对外公开,这样变量就被隐藏起来。这种将对象的变量置于方法保护之下的方式称为封装。例如,可以把飞机抽象为一个对象,用变量来表示它当前的状态,如速度、油量、型号、所处的位置等,它的行为则可以有加速、减速、着陆等。

操作飞机时,不用去考虑飞机内部各个零件是如何运作的细节问题,而只需根据飞机可以做出的行为来使用相应的方法就可以了。实际上,面向对象的程序设计实现了对对

象的封装,而不必关心对象的行为是如何实现的这样一些细节。

通过对对象的封装,能把对其他对象来说并不重要的对象的实现细节隐藏起来,在使用一个对象时,只需要知道怎样引用它的方法而无须知道它的具体实现。这给软件开发人员提供了两个好处。

(1) 模块化。面向对象开发方法很自然地支持了把系统分解成模块的设计原则,一个对象的源代码的便携盒维护独立于其他对象的源代码,而且,对象在系统中容易使用。它是把数据结构和操作这些数据的方法紧密地结合在一起所构成的模块。

(2) 信息隐蔽。每个对象都有一个公共接口使得其他对象可以与其进行通信,但对象能维护自己的私有信息和方法,并能在任何时候在不影响使用它其他对象的情况下改变它们。

这种对象的封装性对创建良好的面向对象的应用程序的开发十分重要。

单独一个对象一般没有什么用处。大的应用程序中通常包含许多对象,并通过这些对象之间的交互来完成一个复杂的功能。例如,在多媒体教室进行教学活动过程中,包括教师对象、学生对象、计算机对象、投影仪对象、音响设备对象和桌子对象等,通过这些对象之间的使用与交互才能顺利完成教学活动。

对象之间的交互通过消息传递来实现。当对象 A 让对象 B 执行它的某个方法时,对象 A 就要像对象 B 发送消息,有时,接收消息的对象需要了解更多信息,以明确该做什么,这时就需要传递一些参数。例如,要使汽车加速,必须发给汽车一个消息,告诉它进行什么样的动作(这里是加速),以及实现这种动作所需要的参数(这里是需要达到的速度等)。

发送消息的实质是要调用接收消息的对象的方法,所以发送消息首先要确定接收消息的对象,然后确定调用对象的方法,最后传递此方法需要的参数。

3. 类

类是具有相似特性(数据元素)和行为(功能)的对象的抽象。描述了该类型所有对象的性质,即统一的状态和行为。用面向对象的术语来说,类定义了所有某种类型的对象的共有的变量和方法。也可以说类的实例是对象,类实际上就是一种数据类型。属于类的某一个对象被称为是类的一个实例,是类的一次实例化结果。从一个类能创建许多对象的实例。类与对象之间的关系可以看成是抽象与具体的关系。例如,你的汽车作为一个具体对象,是汽车类的一个实例。

类具有属性,它是对象状态的抽象,用数据结构来描述类的属性。类具有操作,它是对象行为的抽象,用操作名和实现该操作的方法来描述。

在面向对象的编程中,类形成了一个具有特定功能的模块和一种代码共享手段。它为程序员提供了一种工具,使他们可以方便地建立所需要的对象,这些已有的类还可以使程序员不必重复劳动就可以创造出新的类。

4. 消息和方法

对象之间进行通信的结构称为消息。在对象的操作中,当一个消息发送给某个对象时,消息包含接收对象去执行某种操作的信息。发送一条消息至少要包括说明接受

消息的对象名、发送给该对象的消息名(即对象名、方法名)。一般还要对参数加以说明,参数可以是认识该消息的对象所知道的变量名,或者是所有对象都知道的全局变量名。

类中操作的实现过程称为方法,一个方法有方法名、返回值、参数、方法体。

2.2.2 面向对象的程序设计

1. 面向对象程序设计的发展过程

面向对象出现以前,结构化程序设计是程序设计的主流,结构化程序设计又称为面向过程的程序设计。在面向过程程序设计中,问题被看作一系列需要完成的任务,函数(在此泛指例程、函数、过程)用于完成这些任务。程序=(算法)+(数据结构),即算法是一个独立的整体,数据结构(包含数据类型与数据)也是一个独立的整体。

两者分开设计,以算法或过程为主。这种算法与数据分离,造成一个数据可被多个算法调用,一个算法又可以调用多个数据的局面,这样,一旦产生错误很难跟踪是谁修改了数据。因此要使数据与程序始终保持相容,已经成为程序员一个沉重的负担。尤其开发大型复杂的应用程序时,结构化程序设计更加力不从心。

为了解决上述问题,人们不得不换个角度思考问题。人们发现现实世界本身就是一个对象的世界,任何对象都有一定的属性和操作,也就总能用数据结构与算法两者结合来描述,即算法与数据结构是一个整体,算法总是离不开数据结构,而算法只能是适用于特定的数据结构。这时程序的定律即被改为:对象=(算法+数据结构),程序=(对象+对象+……),于是就产生了新的程序设计方法,即面向对象的程序设计。

面向对象的设计思想是以需要解决的问题空间所涉及的各种对象为主体。对象最主要的特点是以数据为中心,它是集成了数据和对数据操作的一个独立个体。对象的数据封装特性消除了数据与操作分离带来的各种问题,提高了程序的可复用性和可维护性,降低了程序员保持数据与操作相容的负担。

2. 面向对象程序设计的软件开发方法

面向对象编程是思考程序设计时一种新的、与过程化编程全然不同的方式,面向对象的软件开发可概括为以下过程。

(1) 分析用户需求。抽取出存在于用户需求中的各种对象实体、他们之间的相互关系以及他们在整个系统中的位置,从而抽取出对象模型,将用户的需求准确地表达出来。

(2) 根据抽取的模型设计类。包括类中属性的确定、类中方法的功能和实现细节的明确规定以及它们的访问权限,同时还要考虑是否有可以直接引用的已有类等。

(3) 选定一种面向对象的编程语言。具体编码实现上一阶段类的设计,并在开发中引入测试,完善整个解决方案。

由于对象的概念能够更自然地表达和处理现实世界的问题,所以面向对象的软件开发方法可以编写出具有更好的灵活性、可重用性和可扩展性的代码,使得上述“分析、设计、实现”的开发过程更加高效、快捷。即使需要修改原有设计,也能够在以前工作的基础

上从容完成,而不会陷入传统方法中不得不推翻原有设计、重新考虑数据结构和程序结构的尴尬境地。

2.3 C#数据类型

C#是一种简洁、现代、面向对象且类型安全的编程语言。从大的方面来分,C#语言的数据类型可以分为3种,即值类型、引用类型、指针类型。指针类型仅用于非安全代码中。虽然C#是面向对象的语言,然而C#进一步提供了对面向组件编程的支持。本节先来认识一下C#的数据类型。

2.3.1 变量

变量(variable),言外之意即是可变的,用来存储程序所需的数据。

1. 变量的声明


【基本语法】

```
数据类型变量名;
```

其中,变量名必须是有效的标识符。也可以在声明的同时初始化该变量。


【基本语法】

```
数据类型变量名 = 值;
```

其中,除变量名必须是有效的标识符外,值必须与变量声明的数据类型相兼容。

例如,int i;该句声明了一个int(整型)变量i。

如果在一个语句中声明和初始化了多个变量,那么所有的变量都具有相同的数据类型。要声明类型不同的变量,需要使用单独的语句。在多个变量的声明中,不能指定不同的数据类型。

2. 变量初始化

C#编译器需要每个变量在有了初始值之后才能使用该变量。简单地说,C#编译器需要用某个初始值对变量进行初始化,之后才能在操作中引用该变量。大多数现代编译器把没有初始化标记为警告,但C#编译器把它当作错误来看待。这就可以防止无意中从其他程序遗留下来的内存中获取垃圾值。

在C#变量初始化时有两点需要注意。

(1) 首先变量是类或者结构中的字段,如果没有显式地初始化,在默认状态下,创建这些变量时,其初始值就是0。

【例2-1】

```
using System;
namespace gosoa.com {
  class MyFirstClass {
```

```
    static int y;
    static void Main() {
      Console.WriteLine(y);
    }
  }
}
```

在类中声明了一个变量 y，然后输出该变量，编译并运行后会看到输出的结果是 0。

(2) 其次，方法中的变量必须显式地初始化，否则在使用该变量时会出错。

【例 2-2】

```
using System;
namespace gosoa.com {
  class MyFirstClass  {
    static void Main() {
      int y;
      Console.WriteLine(y);

    }
  }
}
```

在编译的时候会报错。需要把 int y 显式地进行初始化才会通过编译。比如初始化 y 的值为 10，即 int y=10；便会通过编译。

3. 变量的作用域

变量的作用域是指可以使用该变量的代码区域。一般情况下，确定作用域有以下规则。

(1) 只要变量所属的类在某个作用域内，其字段(也叫作成员变量)也在该作用域中。

(2) 局部变量存在于声明该变量的块语句或方法结束的大括号之前的作用域。

(3) 在 for、while 循环中声明的变量，只存在于该循环体内。

对于变量的作用域，应注意如下两点。

(1) 局部变量的作用域冲突。

大型程序在不同部分为不同变量使用相同的变量名是很常见的。只要变量的作用域是程序的不同部分，就不会有问题，也不会产生模糊性。但要注意，同名的局部变量不能在同一作用域内声明两次。

(2) 字段和局部变量的作用域冲突。

在某些情况下，可以区分名称相同(尽管其完全限定的名称不同)、作用域相同的两个标识符。此时编译器允许声明第二个变量。原因是 C# 在变量之间有一个基本的区分，它把声明为类型级的变量看作字段，而把在方法中声明的变量看作局部变量。

2.3.2 常量

常量指一经初始化就不会再次被改变的“变量”，在程序的整个运行过程中不允许改变它的值。声明变量时，在变量前面加上 const 关键字就可以把该变量指定为一个常量。在这里需要注意以下两点。

(1) 常量必须在声明时就初始化,而且其赋值后就不能再更改了。

(2) 常量总是静态(static)的,不必在声明常量时添加 static 关键字。

1. 编译时常量



【基本语法】

```
const 数据类型常量名 = 值;
```

编译时常量作为类成员时总是作为 static 成员出现。不允许自己加 static 关键字。编译时常量的值必须是在编译时就能确定下来的,只支持一些基本数据类型。

2. 运行时常量



【基本语法】

```
readonly 数据类型常量名 = 值;
```

运行时常量可以弥补编译时常量不能定义复杂数据类型的缺点。

2.3.3 值类型和引用类型的区别

简单地说,C#中的数据类型可以分为值类型和引用类型。值类型存储在堆栈上,而引用类型存储在管理堆上。在 C#语言中,值类型变量存储的是数据类型所代表的实际数据,值类型变量的值(或实例)存储在栈(Stack)中,赋值语句是传递变量的值。引用类型(如类就是引用类型)的实例也叫对象,不存在栈中,而存储在可管理堆(Managed Heap)中,堆实际上是计算机系统中的空闲内存。

引用类型变量的值存储在栈(Stack)中,但存储的不是引用类型对象,而是存储引用类型对象的引用,即地址,与指针所代表的地址不同,引用所代表的地址不能被修改,也不能转换为其他类型的地址,它是引用型变量,只能引用指定类对象,引用类型变量赋值语句是传递对象的地址。

C#的值类型进一步划分为简单类型(simple type)、枚举类型(enum type)和结构类型(struct type),C#的引用类型进一步划分为类类型(class type)、接口类型(interface type)、数组类型(array type)和委托类型(delegate type)。如表 2-1 所示 C#中的类型系统。

表 2-1 C#类型系统

类型		说明
值类型	简单类型	有符号整型:sbyte、short、int、long
		无符号整型:byte、ushort、uint、ulong
		Unicode 字符:char
		IEEE 浮点型:float、double
		高精度小数:decimal
		布尔型:bool
	结构类型	struct S {...} 形式的用户定义的类型
	枚举类型	enum E {...} 形式的用户定义的类型

续表

类　　型		说　　明
引用类型	类类型	所有其他类型的最终基类：object Unicode
		字符串：string
		class C {...} 形式的用户定义的类型
	接口类型	interface I {...} 形式的用户定义的类型
	数组类型	一维和多维数组，如 int[]和 int[,]
	委托类型	delegate T D(...) 形式的用户定义的类型

2.3.4　值类型变量

C#语言值类型可以分为以下几种。

(1) 简单类型(simple types)。包括数值类型和布尔类型(bool)。数值类型又细分为整数类型、字符类型(char)、浮点数类型和十进制类型(decimal)。

(2) 结构类型(struct types)。

(3) 枚举类型(enumeration types)。

C#语言值类型变量无论如何定义，总是值类型变量，不会变为引用类型变量。

1. 简单类型

简单类型也是结构类型，因此有构造函数、数据成员、方法、属性等。即使一个常量，C#也会生成结构类型的实例，因此也可以使用结构类型的方法。简单类型如表 2-2 所示，包括整数类型、字符类型、布尔类型、浮点数类型、十进制类型(decimal)。

表 2-2　简单类型

类型	保留字	System 命名空间中的名字	字节数	取 值 范 围	说　　明
整数类型	sbyte	System. Sbyte	1	－128～127	8 位有符号整数
	byte	System. Byte	1	0～255	8 位无符号整数
	short	System. Int16	2	－32768～32767	16 位有符号整数
	ushort	System. UInt16	2	0～65535	16 位无符号整数
	int	System. Int32	4	－2147483648～2147483647	32 位有符号整数
	uint	System. UInt32	4	0～4292967295	32 位无符号整数
	long	System. Int64	8	-2^{31}～$2^{31}-1$	64 位有符号整数
	ulong	System. UInt64	8	0～$2^{64}-1$	64 位无符号整数
字符类型	char	System. Char	2	0～65535	• 一个 16 位的 unicode 字符 • 使用单引号括起来。双引号括起来的是字符串类型的
浮点类型	float	System. Single	4	3.4E－38～3.4E＋38	32 位单精度浮点数
	double	System. Double	8	1.7E－308～1.7E＋308	64 位双精度浮点数
布尔类型	bool	System. Boolean		(true,false)	包含 true 和 false
decimal 类型	decimal	System. Decimal	16	±1.0～7.9	一种财务专用数据类型，是 128 位高精度十进制表示法

C#简单类型使用方法和C、C++中相应的数据类型基本一致。需要注意以下几点。

(1) 无论在何种系统中,C#每种数据类型所占字节数是一定的。

(2) 整数类型不能隐式被转换为字符类型(char),如 char c1=10 是错误的,必须写成:char c1=(char)10,char c='A',char c='\x0032';char c='\u0032'。

(3) 布尔类型有两个值:false、true。不能认为整数 0 是 false,其他值是 true。例如,bool x=1 是错误的,不存在这种写法,只能写成 x=true 或 x=false。

(4) 十进制类型(decimal)也是浮点数类型,只是精度比较高,一般用于财政金融计算。

2. 结构类型

结构类型和类一样,可以声明构造函数、数据成员、方法、属性等,是能够包含数据成员和函数成员的数据结构。结构类型的变量直接存储该结构的数据。

结构和类的最根本的区别是结构为值类型,类为引用类型。和类不同,结构不能从另外一个结构或者类派生,其本身也不能被继承,因此不能定义抽象结构,结构成员也不能被访问权限控制字 protected 修饰,也不能用 virtual 和 abstract 修饰结构方法。在结构中不能定义析构函数。所有结构类型都隐式地从类型 System. ValueType 继承。System. ValueType 继承自 System. Object。

虽然结构不能从类和结构派生,但是结构能够继承接口,结构继承接口的方法和类继承接口的方法基本一致。

**【基本语法】**

```
Struct 结构名称
{   …   }
```

【例 2-3】 定义一个点结构 point。

```
using System;
structpoint                      //结构定义
{   public int x,y;              //结构中也可以声明构造函数和方法,变量不能赋初值
}
class Test                       //类定义
{   static void Main()
    {   pointP1;
        P1.x = 166;
        P1.y = 111;
        point P2;
        P2 = P1;                 //值传递,使 P2.x = 166,P2.y = 111
        point P3 = new point();  //用 new 生成结构变量 P3,P3 仍为值类型变量
    }                        //用 new 生成结构变量 P3 仅表示调用默认构造函数,使 x = y == 0
}
```

3. 枚举类型

枚举(enum)是具有一组命名常量的独特的值(结构)类型。每个枚举类型都有一个相应的整型类型,称为该枚举类型的基础类型(underlying type)。没有显式声明基础类

型的枚举类型所对应的基础类型是 int。枚举类型的存储格式和取值范围由其基础类型确定。所有枚举类型默认继承自 System. Enum 类型，System. Enum 继承自 System. ValueType。

【例 2-4】 定义星期。

```
using System;
class Class1
{   enum Days {Sun, Mon, Tue, Wed, Thu, Fri,Sat};
    //使用 Visual Studio.Net,enum 语句添加在[STAThread]前边
    static void Main(string[ ] args)
    {   Days day = Days.Tue;
        int x = (int)Days.Tue;//x = 2
        Console.WriteLine("day = {0},x = {1}",day,x);
     //显示结果为:day = Tue,x = 4
    }
}
```

在此枚举类型 Days 中，每个元素的默认类型为 int，其中 Sun＝0，Mon＝1，Tue＝2，以此类推。也可以直接给枚举元素赋值。例如：

```
enum Days{Sun = 0,Mon = 1,Tue = 2,Wed = 3,Thu = 4,Fri = 5,Sat = 6};
```

在此枚举中，Sun＝0，Mon＝1，Tue＝2，Wed＝3 等。C＃枚举元素类型可以是 byte、sbyte、short、ushort、int、uint、long 和 ulong 类型，但不能是 char 类型。例如：

```
enum Days:byte{Sun,Mon,Tue,Wed,Thu,Fri,Sat};//元素为字节类型
```

4. 值类型的初值和默认构造函数

所有变量都要求必须有初值，如没有赋值，则采用默认值。对于简单类型，如 sbyte、byte、short、ushort、int、uint、long 和 ulong 默认值为 0，char 类型默认值是(char)0，float 为 0.0f，double 为 0.0d，decimal 为 0.0m，bool 为 false，枚举类型为 0，在结构类型和类中，数据成员的数值类型变量设置为默认值，引用类型变量设置为 null。

可以显式地赋值，如 int i＝0。而对于复杂结构类型，其中的每个数据成员都按此种方法赋值，显得过于麻烦。由于数值类型都是结构类型，可用 new 语句调用其构造函数初始化数值类型变量，如 int j＝new int()。



【小提示】

用 new 语句并不是把 int 变量变为引用变量，j 仍是值类型变量，这里 new 仅仅是调用其构造函数。所有数值类型都有默认的无参数的构造函数，其功能就是为该数值类型赋初值为默认值。对于自定义结构类型，由于已有默认的无参数的构造函数，因此不能再定义无参数的构造函数，但可以定义有参数的构造函数。

2.3.5 引用类型

C＃ 的引用类型进一步划分为类类型 (class type)、接口类型 (interface type)、数组

类型（array type）和委托类型（delegate type）。

C#语言中引用类型可以分为以下几种。

(1) 类：C#语言中预定义了一些类，如对象类(object 类)、字符串类等。当然，程序员可以定义其他类。

(2) 接口。

(3) 数组。

(4) 委托。

C#语言引用类型变量无论如何定义，总是引用类型变量，不会变为值类型变量。C#语言引用类型对象一般用运算符 new 建立，用引用类型变量引用该对象。

1. 类

1) C#中类的定义

类可以认为是对结构的扩充，C#中类不但可以包括数据，还包括处理这些数据的函数。类是对数据和处理数据的方法(函数)的封装。类是对某一类具有相同特性和行为的事物的描述。



【基本语法】

属性 类修饰符 class 类名{类体}

【语法说明】

(1) class 是保留字，表示定义一个类，类名是定义类的名称。

(2) 关键字 class、类名和类体是必需的，其他项是可选项。

(3) 类修饰符包括 new、public、protected、internal、private、abstract 和 sealed。

(4) 类体用于定义类的成员。

【例 2-5】 定义一个描述个人情况的类 Person。

```
using System;
class Person                           //类的定义,class 是保留字,表示定义一个类,Person 是类名
{   private string name = "张三";               //类的数据成员声明
    private int age = 12;                      //private 表示私有数据成员
    public void Display()                      //类的方法(函数)声明,显示姓名和年龄
    {   Console.WriteLine("姓名:{0},年龄: {1}",name,age);
    }
    public voidSetName(string PersonName)      //修改姓名的方法(函数)
    {   name = PersonName;
    }
    public voidSetAge(int PersonAge)
    {   age = PersonAge;
    }
}
```

Console. WriteLine("姓名:{0},年龄: {1}",name,age)的意义是将第二个参数变量 name 变为字符串填到{0}位置，将第三个参数变量 age 变为字符串填到{1}位置，将第一

个参数表示的字符串在显示器上输出。


【小提示】

这里实际定义了一个新的数据类型，为用户自己定义的数据类型，是对个人的特性和行为的描述，它的类型名为 person，和 int、char 等一样为一种数据类型。用定义新数据类型 person 类的方法把数据和处理数据的函数封装起来。

2）类成员的存取控制

一般希望类中一些数据不被随意修改，仅按指定方法修改，即隐蔽一些数据。同样一些函数也不希望被其他类程序调用，只能在类内部使用。可用访问权限控制字来解决这个问题。常用的访问权限控制字如下：private（私有）、public（公有）。在数据成员或函数成员前增加访问权限控制字，可以指定该数据成员或函数成员的访问权限。

私有数据成员只能被类内部的函数使用和修改，私有函数成员只能被类内部的其他函数调用。类的公有函数成员可以被类的外部程序调用，类的公有数据成员可以被类的外部程序直接使用修改。公有函数是一个类和外部通信的接口，外部函数通过调用公有函数，按照预先设定好的方法修改类的私有成员。对于上述例子，name 和 age 是私有数据成员，只能通过公有函数 SetName()和 SetAge()修改，即它们只能按指定方法修改。


【小提示】

这里再一次地体现了封装的意义，第一是把数据和处理数据的方法同时定义在类中；第二是用访问权限控制字使数据隐蔽。

3）类的对象

person 类仅是一个用户新定义的数据类型，由它可以生成 person 类的实例，C# 语言叫对象。用以下方法声明类的对象。

```
personOnePerson = new Person();
```

此语句的意义是建立 person 类对象，返回对象地址赋值给 person 类变量 OnePerson。也可以分两步创建 person 类的对象：

```
PersonOnePerson; OnePerson = new Person();
```

OnePerson 虽然存储的是 person 类对象地址，但不是 C 中的指针，不能像指针那样可以进行加减运算，也不能转换为其他类型地址，它是引用型变量，只能引用（代表）person 对象，具体意义参见以后章节。和 C、C++不同，C# 只能用此种方法生成类对象。

在程序中，可以用 OnePerson. 方法名或 OnePerson. 数据成员名访问对象的成员，如 OnePerson. Display()，公用数据成员也可以这样访问。


【小提示】

C# 语言中不包括 C++语言中的—>符号。

4）类的构造函数和析构函数

在建立类的对象时，需做一些初始化工作，如对数据成员初始化。这些可以用构造函

数来完成。每当用 new 生成类的对象时，自动调用类的构造函数。因此，可以把初始化的工作放到构造函数中完成。构造函数和类名相同，没有返回值。例如，可以定义 person 类的构造函数。

【例 2-6】

```
public Person(string Name, int Age)            //类的构造函数，函数名和类同名，无返回值
{   name = Name;
    age = Age;
}
```

当用 PersonOnePerson＝new Person("张五"，20)语句生成 person 类对象时，将自动调用以上构造函数。



【小提示】

变量和类的对象都有生命周期，生命周期结束，这些变量和对象就要被撤销。类的对象被撤销时，将自动调用析构函数。一些善后工作可放在析构函数中完成。析构函数的名字为类名，无返回类型，也无参数。person 类的析构函数为 person()。C＃中类析构函数不能显式地被调用，它是被垃圾收集器撤销不被使用对象时自动调用的。

5）类的构造函数的重载

在 C＃语言中，同一个类中的函数，如果函数名相同，而参数类型或个数不同，认为是不同的函数，这叫函数重载。仅返回值不同，不能看作不同的函数。这样，可以在类定义中，定义多个构造函数，名字相同，参数类型或个数不同。根据生成类的对象方法不同，调用不同的构造函数。例如，可以定义 person 类没有参数的构造函数。

【例 2-7】

```
public Person()                                //类的构造函数，函数名和类同名，无返回值
{   name = "张三";
    age = 12;
}
```

用语句 Person OnePerson＝new Person("李四"，30)生成对象时，将调用有参数的构造函数，而用语句 Person OnePerson＝new Person()生成对象时，调用无参数的构造函数。由于析构函数无参数，因此，析构函数不能重载。

6）对象类(object 类)

C＃中的所有类型(包括数值类型)都直接或间接地以 object 类为基类。对象类(object 类)是所有其他类的基类。任何一个类定义，如果不指定基类，默认 object 为基类。C＃语言规定，基类的引用变量可以引用派生类的对象，因此，对一个 object 的变量可以赋予任何类型的值，例如：

```
int x = 25;
object obj1;
obj1 = x;
object obj2 = 'A';
```

object 关键字是在命名空间 System 中定义的，是类 System. Object 的别名。

【小提示】

派生类的引用变量不可以引用基类的对象。

7）字符串类（string 类）

C＃还定义了一个基本的类 string，专门用于对字符串的操作。这个类也是在名字空间 System 中定义的，是类 System. String 的别名。利用 String 类可以进行字符串的创建、截取、替换及合并等操作，也可以用“＋”方便地进行字符串的合并。

【小提示】

大写 String 与小写 string 是完全相同的，大写是指. NET 类库中的 String 类型，小写是 C＃关键字，也是对应到 String 这个类型上去的，如在 C＃中 int 和 Int32 也是这样对应的。

2. 数组类

数组（array）是一种包含若干变量的数据结构，这些变量都可以通过计算索引进行访问。数组中包含的变量（元素（element））具有相同的类型，该类型称为数组的元素类型（element type）。

数组类型为引用类型，因此数组变量的声明只是为数组实例的引用留出空间。在运行时使用 new 运算符动态创建（须指定长度），长度在该实例的生存期内是固定不变的。数组元素的索引范围从 0 到 Length－1。new 运算符自动将数组的元素初始化为它们的默认值，如将所有数值类型初始化为零、将所有引用类型初始化为 null。

在进行批量处理数据时，要用到数组。数组按照数组名、数据元素的类型和维数来进行描述。C＃语言中数组是类 System. Array 对象。例如，声明一个整型数数组：int[] arr＝new int[5]；实际上生成了一个数组类对象，arr 是这个对象的引用（地址）。

在 C＃中数组可以是一维的也可以是多维的，同样也支持数组的数组，即数组的元素还是数组。一维数组最为普遍，用得也最多。数组下标一般是从 0 开始，也提供了其他方式支持非从 0 下标开始的数组。

1）数组定义

【基本语法】

```
类型[]数组名;
```

在声明数组时，应先定义数组中元素的类型，其后是一个空方括号和一个变量名。例如，下面声明了一个包含整型元素的数组：

```
int[] myArray;
```

2）数组初始化

声明了数组后，就必须为数组分配内存，以保存数组的所有元素。数组是引用类型，所以必须给它分配堆上的内存。为此，应使用 new 运算符，指定数组中元素的类型和数

量来初始化数组的变量。例如，为前面定义的 myArray 整型数组进行初始化：

```
myArray = new int[n];
```



【小提示】

可以直接用 new 运算符建立一个包含 n 个整型元素的一维数组：

```
int[]myArray = new int[n];
```

在指定了数组的大小后，如果不复制数组中的所有元素，就不能重新设置数组的大小。

使用 C# 编译器还有一种更简化的形式。使用花括号可以同时声明和初始化数组，编译器生成的代码与前面相同，例如：

```
int[] myArray = {4, 7, 11, 2};
```

3）访问数组元素

数组在声明和初始化后，就可以使用索引器访问其中的元素了。数组只支持有整型参数的索引器。



【基本语法】

```
array[索引];
```

通过索引器传送元素号，就可以访问数组。索引器总是以 0 开头，表示第一个元素。可以传送给索引器的最大值是元素个数减 1。



【小提示】

如果使用错误的索引器值(不存在对应的元素)，就会显示 IndexOutOfRangeException 类型异常的提示信息。

【例 2-8】 一个一维数组的例子。

```
using System;
class Test
{static void Main()
    {
       int[] arr = new int[3];               //用 new 运算符建立一个 3 个元素的一维数组
       for(int i = 0;i < arr.Length;i++)    //arr.Length 是数组类变量,表示数组元素个数
       arr[i] = i * i;                       //数组元素赋初值,arr[i]表示第 i 个元素的值
       for (int i = 0;i < arr.Length;i++)   //数组第一个元素的下标为 0
       Console.WriteLine("arr[{0}] = {1}",i,arr[i]);
    }
}
```

这个程序创建了一个 int 类型 3 个元素的一维数组，初始化后逐项输出。其中 arr.Length 表示数组元素的个数。注意数组定义不能写为 C 语言格式：int arr[]。

程序的输出为：

```
arr[0] = 0
arr[1] = 1
arr[2] = 4
```

4）多维数组

【例 2-9】 基础介绍多维数组。

```
string[] a1;                              //一维 string 数组类引用变量 a1
string[,] a2;                             //二维 string 数组类引用变量 a2
a2 = new string[2,3];
a2[1,2] = "abc";
string[,,] a3;                            //三维 string 数组类引用变量 a3
string[][] j2;                            //数组的数组,即数组的元素还是数组
string[][][][] j3;
```

在数组声明时,可以对数组元素进行赋值。看下面的例子:

```
int[] a1 = new int[]{1,2,3};              //一维数组,有 3 个元素
int[] a2 = new int[3]{1,2,3};             //此格式也正确
int[] a3 = {1,2,3};                       //相当于 int[] a3 = new int[]{1,2,3};
int[,] a4 = new int[,]{{1,2,3},{4,5,6}};  //二维数组,a4[1,1] = 5
int[][] j2 = new int[3][];                //定义数组 j2,有 3 个元素,每个元素都是一个数组
j2[0] = new int[]{1,2,3};                 //定义第一个元素,是一个数组
j2[1] = new int[]{1, 2, 3, 4, 5, 6};      //每个元素的数组可以不等长
j2[2] = new int[]{1, 2, 3, 4, 5, 6, 7, 8, 9};
```

3. 委托

委托(delegate)是 C# 中的一种类型,它实际上是一个能够持有对某个方法的引用的类。与其他的类不同,delegate 类能够拥有一个签名(signature),并且它只能持有与它的签名相匹配的方法的引用。委托类型(delegate type)表示对具有特定参数列表和返回类型的方法的引用。通过委托能够将方法作为实体赋值给变量和作为参数传递。

委托类似于在其他某些语言中的函数指针的概念,但是与函数指针不同,委托是面向对象的,并且是类型安全的。

委托声明定义一个从 System.Delegate 类派生的类。委托实例封装了一个调用列表,该列表列出了一个或多个方法,每个方法称为一个可调用实体。对于实例方法,可调用实体由该方法和一个相关联的实例组成。对于静态方法,可调用实体仅由一个方法组成。用一个适当的参数集来调用一个委托实例,就是用此给定的参数集来调用该委托实例的每个可调用实体。

1）实现委托

**【基本语法】**

```
delegate void MyDelegate(变量);
```

实现一个 delegate 是很简单的,通过以下 3 个步骤即可实现一个 delegate。

（1）声明一个 delegate 对象，它应当与你想要传递的方法具有相同的参数和返回值类型。

（2）创建 delegate 对象，并将你想要传递的函数作为参数传入。

（3）在要实现异步调用的地方，通过上一步创建的对象来调用方法。

与其他类型相比较，委托将方法作为参数传递。通常传递的是变量(字段)，委托则是传递方法。可以进行多路广播，可以同时维持多个方法的引用。回调方法时，底层代码定义方法签名的类型(委托)，定义委托成员；上层代码创建方法，创建委托实例，让需要调用的方法传给底层；底层通过调用委托，调用上层方法。需要注意的是，即使函数签名相同，即便同时定义 DelegateA da；DelegateB db；也不能执行 da=db；而且委托类型都是密封的(sealed)，不能继承。

【例 2-10】

```
using System;
public class MyDelegateTest
{                                              // 步骤 1,声明 delegate 对象
  public delegate void MyDelegate(string name);
  // 这是欲传递的方法,它与 MyDelegate 具有相同的参数和返回值类型
  public static void MyDelegateFunc(string name)
  {
    Console.WriteLine("Hello, {0}", name);
  }
     public static void Main()
     {                                         // 步骤 2,创建 delegate 对象
       MyDelegate md  =  new MyDelegate(MyDelegateTest.MyDelegateFunc);
                                               // 步骤 3,调用 delegate
      md("sam1111");
     }
}
```

输出结果是：Hello，sam1111

2）重要成员

委托包含 Target、Method、Invoke、BeginInvoke、EndInvoke 等重要成员。

（1）Target：object 类型的属性，指向回调函数所属的对象实例(对于实例方法而言)。引用的方法是静态方法时，Target 为 null。

（2）Method：System.Reflection.MethodInfo 类型的属性，指向回调函数。

（3）Invoke：函数，同步执行委托。

（4）BeginInvoke：开始异步执行委托。

（5）EndInvoke：完成异步执行。

3）运算操作


【基本语法】

```
myDelegate  + =  new MyDelegate(AddNumber.add2);
```

【语法说明】

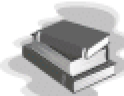
(1) 将一个委托 A 与另一个委托 B 连接,将连接后的新委托再赋给原委托 A。

(2) 实质是使用的 System. Delegate 的静态方法 Combine:

```
myDelegate = (MyDelegate)Delegate.Combine(myDelegate,new MyDelegate(AddNumber.add2));
```

【基本语法】

```
myDelegate -= new MyDelegate(AddNumber.add2);
```

【语法说明】

(1) 一个委托 A 的调用列表中移除另一个委托 B 的最后一个调用列表,将移除后的新委托再赋给原委托 A。

(2) 实质是使用的 System. Delegate 的静态方法 Remove:

```
myDelegate = (MyDelegate)Delegate.Remove(myDelegate,new MyDelegate(AddNumber.add2));
```

4. 接口

接口(interface)是用来定义一种程序的协定。实现接口的类或者结构要与接口的定义严格一致。C#定义接口,里面包含方法,但没有方法具体实现的代码,然后在继承该接口的类里面要实现接口的所有方法的代码。接口其实就是类和类之间的一种协定、一种约束。

使用 interface 来定义接口,下面示例定义了一个接口。

```
public interface IBark
{
  void Bark();
}
```

再定义一个类,继承于 IBark,并且必须实现其中的 Bark()方法。

```
public class Cat:IBark {
  public Cat()  {}
  public void Bark()
  {
      Consol.write("喵喵");
  }
}
```

然后,声明 Cat 的一个实例,并调用 Bark()方法。

```
Cat 招财 = new Cat();
  招财.Bark();
```

试想一下,若想调用 Bark()方法,只需要在 Dog()中声明这样的一个方法不就行了吗,为什么还要用接口呢?其实原因很简单,因为接口中并没有 Bark()具体实现,真的实

现还是要在 Cat()中。所有继承了 IBark 接口的类中必须实现 Bark()方法。那么从用户(使用类的用户)的角度来说,如果他知道了某个类是继承于 IBark 接口,那么他就可以放心大胆地调用 Bark()方法,而不用管 Bark()方法具体是如何实现的。

2.3.6　类型转换

在编写 C# 语言程序中,经常会碰到类型转换问题。例如,整型数和浮点数相加,C# 会进行隐式转换。作为程序员应记住类型转换的一些基本原则,编译器在转换发生问题时,会给出提示。C# 语言中类型转换分为隐式转换、显式转换、加框(boxing)和消框(unboxing)3 种。

1. 隐式转换

隐式转换就是系统默认的、不需要加以声明就可以进行的转换,如从 int 类型转换到 long 类型就是一种隐式转换。在隐式转换过程中,转换一般不会失败,转换过程中也不会导致信息丢失。例如:

```
int i = 10;
long l = i;
```

2. 显式转换

显式类型转换,又叫强制类型转换。与隐式转换正好相反,显式转换需要明确地指定转换类型,显式转换可能导致信息丢失。下面的例子把长整型变量显式转换为整型。

```
long l = 5000;
int i = (int)l;                              //如果超过 int 取值范围,将产生异常
```

3. 加框和消框

加框(boxing)和消框(unboxing)是 C# 语言类型系统提出的核心概念,加框是值类型转换为 object(对象)类型,消框是 object(对象)类型转换为值类型。有了加框和消框的概念,对任何类型的变量来说最终都可以看作是 object 类型。

1) 加框操作

把一个值类型变量加框也就是创建一个 object 对象,并将这个值类型变量的值赋值给这个 object 对象。例如:

```
int i = 10;
object obj = i;                              //隐式加框操作,obj 为创建的 object 对象的
引用
```

也可以用显式的方法来进行加框操作,例如:

```
int i  = 10;
object obj = object(i);                      //显式加框操作
```

值类型的值加框后,值类型变量的值不变,仅将这个值类型变量的值赋值给这个 object 对象。看下面的程序。

```
using System
class Test
{   public static void Main()
   {   int n = 200;
       object o = n;
       o = 201;                              //不能改变 n
       Console.WriteLine("{0},{1}",n,o);
   }
}
```

输出结果为：200,201。这就证明了值类型变量 n 和 object 类对象 o 都是独立存在的。

2）消框操作

和加框操作正好相反，消框操作是指将一个对象类型显式地转换成一个值类型。消框的过程分为两步：首先检查这个 object 对象，看它是否为给定的值类型的加框值，如是，则把这个对象的值复制给值类型的变量。举个例子来看看一个对象消框的过程。

```
int i = 10;
object obj = i;
int j = (int)obj;                            //消框操作
```

可以看出消框过程正好是加框过程的逆过程，要注意加框操作和消框操作必须遵循类型兼容的原则。

3）加框和消框的使用

定义以下函数。

```
void Display(Object o)                       //注意,o 为 Object 类型
{   int x = (int)o;                          //消框
    System.Console.WriteLine("{0},{1}",x,o);
}
```

调用此函数：

```
int y = 20;Display(y);
```

在此利用了加框概念，虚参被实参替换：Object o＝y，也就是说，函数的参数是 Object 类型，可以将任意类型实参传递给函数。

2.4 C＃运算符

2.4.1 运算符分类

运算符（operator）是一种类成员，它定义了可应用于类实例的特定表达式运算符的含义。如果按照运算符所作用的操作数个数来分，C＃语言的运算符可以分为以下几种类型。

（1）一元运算符：一元运算符作用于一个操作数，如－X、＋＋X、X－－等。

（2）二元运算符：二元运算符对两个操作数进行运算，如 x＋y。

(3) 三元运算符：三元运算符只有一个：x?y:z。

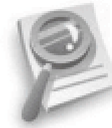
C#语言运算符的详细分类及操作符从高到低的优先级顺序见表2-3。

表2-3 C#运算符

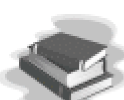
类　别	操　作　符
初级运算符	(x)、x.y、f(x)、a[x]、x++、x--、new type of sizeof checked unchecked
一元运算符	+、-、!、~、++x、-x、(T)x
乘除运算符	*、/、%
加减运算符	+、-
移位运算符	<、<>、>
关系运算符	<>、<=、>=、is as
等式运算符	==、!=
逻辑与运算符	&
逻辑异或运算符	^
逻辑或运算符	|
条件与运算符	&&
条件或运算符	||
条件运算符	?:
赋值运算符	=、*=、/=、%=、+=、-=、<<=、>>=、&=、^=、|=

2.4.2 测试运算符 is

is运算符用于动态地检查表达式是否为指定类型。

**【基本语法】**

```
e isT;
```

**【语法说明】**

e是一个表达式，T是一个类型，该式判断e是否为T类型，返回值是一个布尔值。

【例2-11】 请参看资源包中的示例文件：2-10.cs。

2.4.3 typeof运算符

typeof操作符用于获得指定类型在system名字空间中定义的类型名字。

**【基本语法】**

```
typeof(T);
```

**【语法说明】**

T是一个类型，返回值是指定类型在system名字空间中定义的类型名字。

【例 2-12】 请参看资源包中的示例文件：2-11.cs。

2.4.4 溢出检查运算符 checked 和 unchecked

在进行整型算术运算(如+、-、*、/等)或从一种整型显式转换到另一种整型时，有可能出现运算结果超出这个结果所属类型值域的情况，这种情况称为溢出。整型算术运算表达式可以用 checked 或 unchecked 溢出检查操作符，决定在编译和运行时是否对表达式溢出进行检查。

如果表达式不使用溢出检查操作符或使用了 checked 操作符，常量表达式溢出，在编译时将产生错误，表达式中包含变量，程序运行时执行该表达式产生溢出，将产生异常提示信息。而使用了 unchecked 操作符的表达式语句，即使表达式产生溢出，编译和运行时也不会产生错误提示。但这往往会出现一些不可预期的结果，所以要小心使用 unchecked 运算符。

下面用例 2-13 来说明 checked 和 unchecked 运算符的使用方法。

【例 2-13】

```
using System;
class Class1
{   static void Main(string[ ] args)
    {   const int x = int.MaxValue;
        unchecked                                   //不检查溢出
        {   int z = x * 2;                          //编译时不产生编译错误,z = -2
            Console.WriteLine("z = {0}",z);         //显示 -2
        }
        checked                                     //检查溢出
        {   int z1 = (x * 2);                       //编译时会产生编译错误
            Console.WriteLine("z = {0}",z1);
        }
    }
}
```

2.4.5 new 运算符

new 运算符可以创建值类型变量、引用类型对象，同时自动调用构造函数。举例如下。

(1) 用 new 创建整型变量 x，调用默认构造函数：int x=new int()；。

(2) 用 new 建立 person 类对象：PersonC1=new Person ()；//Person 变量是 C1 对象的引用。

(3) 数组也是类，创建数组类对象：int[] arr=new int[2]；//arr 是数组对象的引用。



【小提示】

需要注意的是，int x=new int()语句将自动调用 int 结构不带参数的构造函数，给 x 赋初值 0，x 仍是值类型变量，不会变为引用类型变量。

2.5 C#控制语句

C#语言控制语句主要分为选择语句、循环语句及异常处理语句。包括 if 语句、switch 语句、while 语句、do...while 语句、for 语句、foreach 语句、break 语句、continue 语句、return 语句、异常处理语句等，其中 foreach 语句和异常处理语句是 C#语言新增加的控制语句。

2.5.1 选择语句

当运用选择语句时，即定义了一个控制语句，它的值控制了哪个语句被执行。在 C#中用到两个选择语句，即 if 语句和 switch 语句。

1. if 语句

最常用到的语句是 if 语句。该语句是否被执行取决于布尔表达式。



【基本语法】

```
If (布尔条件)  {语句...}
```

当然，也可以有 else 分支，当布尔表达式的值为假时，该分支就被执行：



【基本语法】

```
If (布尔表条件)  {语句 1...}
Else  {语句 2...}
```

或者

```
If (布尔条件 1)  {语句 1...}
Else if (布尔条件 2) {语句 2...}
```



【小提示】

在 C#中 if 语句仅允许布尔(bool)数据类型的结果。

【例 2-14】 请参看资源包中的示例文件：2-14.cs。

2. switch 语句

和 if 语句相比，switch 语句有一个控制表达式，而且语句按它们所关联的控制表达式的常量运行。switch case 是多分支选择语句，用来实现多分支选择结构。适合于从一组互斥的分支中选择一个来执行。类似于 if 语句，但 switch 语句可以一次将变量与多个值进行比较，而不是仅比较一个。switch 参数后面跟一组 case 子句，如果 switch 参数中的值与某一个 case 后面的判断式相等，就执行 case 子句中的代码。

执行完后用 break 语句标记每个 case 代码的结尾，跳出 switch 语句；也可在 switch 语句中包含一个 default 语句，当所有 case 中的常量表达式的值都没有与 switch 中表达式的值相等，就执行 default 子句中的代码。default 子句可有可无，一个 switch 语句中有

且仅有一个 default 分支。case 后的值必须是常量表达式,不允许使用变量。case 子句的排放顺序无关紧要;default 子句也可放到最前;任何两个 case 的值不能相同。

【基本语法】

```
switch(控制表达式)
{
    case 常量表达式 1: 语句 1
    case 常量表达式 2: 语句 2
     ⋮
    default: 语句 n
}
```

【语法说明】

(1) 控制表达式求值。

(2) 如果 case 标签后的常量表达式符合控制语句所求出的值,内含语句被执行。

(3) 如果没有常量表达式符合控制语句,在 default 标签内的内含语句被执行。

(4) 如果没有一个符合 case 标签,且没有 default 标签,控制转向 switch 语段的结束端。

【例 2-15】 请参看资源包中的示例文件:2-15.cs。

另外,在 switch 语句中可以使用 goto 标签和 gotodefault 语句来实现程序的跳转。

2.5.2 循环语句

当想重复执行某些语句或语句段时,依据当前不同的任务,C# 提供了 4 个不同的循环语句以供使用,如 for 语句、foreach 语句、while 语句和 do while 语句。

1. for 语句

for 语句用来依据特定条件来多次重复执行某些代码。

【基本语法】

```
for(初始化; 条件表达式; 结束一次循环的后续操作)
{
  循环语句
}
//后续代码
```

【语法说明】

控制表达式求值初始化、条件和循环都是可选的。

如果忽略了条件,就会产生一个死循环,要用到跳转语句(break 或 goto)才能退出。例如:

```
for(;;) {
```

```
        break;                               //由于某些原因
    }
```

2. foreach 语句

foreach 语句是 C# 语言新引入的语句，C 和 C++ 中没有这个语句，它借用 Visual Basic 中的 foreach 语句。foreach 语句用于枚举一个集合的元素。与 for 语句相比有更简洁的语法。

【基本语法】

```
foreach (变量类型变量名 in 表达式)
{
  循环语句
}
//后续代码
```

【语法说明】

表达式必须是一个数组或其他集合类型，每一次循环从数组或其他集合中逐一取出数据，赋值给指定类型的变量，该变量可以在循环语句中使用、处理，但不允许修改变量，该变量的指定类型必须和表达式所代表的数组或其他集合中的数据类型一致。

```
using System;
class Test()
{   public static void Main()
    {   int[] list = {10,20,30,40};              //数组
        foreach(int m in list)
        Console.WriteLine("{0}",m);
    }
}
```

对于一维数组，foreach 语句循环顺序是从下标为 0 的元素开始一直到数组的最后一个元素。对于多维数组，元素下标的递增是从最右边那一维开始的。同样 break 和 continue 可以出现在 foreach 语句中，功能不变。

3. while 语句

【基本语法】

```
while(条件)                                   //这个条件为布尔表达式
{
    循环体语句;
}
```

【语法说明】

条件语句是一个布尔表达式，控制循环体语句被执行的次数。可以使用 break 和 continue 语句来控制 while 语句中的执行语句。它的运行方式同在 for 语句中的完全相同。

【例 2-16】 请参看资源包中的示例文件：2-16. cs。

4. do while 语句

**【基本语法】**

```
do
{
    //循环体语句;
}while(测试条件);
```

【语法说明】

先执行循环体语句，然后测试 while 中的条件，如果测试条件为 true，就再次执行循环体语句，直到测试结果为 false 时就退出循环。

【例 2-17】 请参看资源包中的示例文件：2-17. cs。

**【小提示】**

break 语句退出直接封闭它的 switch、while、do while 或 for 语句。当多个 switch、while、do while 或 for 语句彼此嵌套时，break 语句只应用于最里层的语句。直接跳出当前循环。

continue 语句开始直接封闭它的 while、do while 或 for 语句的一次新迭代。进入下一次循环。当多个 while、do while 或 for 语句互相嵌套时，continue 语句只应用于最里层的语句。

return 语句将控制返回到出现 return 语句的函数成员的调用方，结束当前方法，跳转回到调用位置。不带表达式的 return 语句只能用在不计算值的函数成员中，即只能用在返回类型为 void 的方法、属性或索引器的 set 访问器、事件的 add 和 remove 访问器、实例构造函数、静态构造函数或析构函数中。

带表达式的 return 语句只能用在计算值的函数成员中，即返回类型为非 void 的方法、属性或索引器的 get 访问器或用户定义的运算符。必须存在一个隐式转换，使得该表达式兼容于包含它的函数的返回类型。

2.5.3 异常语句

在编写程序时，不仅要关心程序的正常操作，还应该考虑到程序运行时可能发生的各类不可预期的事件，如用户输入错误、内存不够、磁盘出错、网络资源不可用、数据库无法使用等，所有这些错误被称为异常，不能因为这些异常使程序运行产生问题。各种程序设计语言经常采用异常处理语句来解决这类异常问题。

C＃提供了一种处理系统级错误和应用程序级错误的结构化的、统一的、类型安全的方法。C＃异常语句包含 try 子句、catch 子句和 finally 子句。try 子句中包含可能产生异常的语句，该子句自动捕捉执行这些语句过程中发生的异常。catch 子句中包含了对不同异常的处理代码，可以包含多个 catch 子句，每个 catch 子句中包含了一个异常类型，这个异常

类型必须是 System.Exception 类或它的派生类引用变量,该语句只捕捉该类型的异常。

可以有一个通用异常类型的 catch 子句,该 catch 子句一般在事先不能确定会发生什么样的异常的情况下使用,也就是可以捕捉任意类型的异常。一个异常语句中只能有一个通用异常类型的 catch 子句,而且如果有的话,该 catch 子句必须排在其他 catch 子句的后面。无论是否产生异常,子句 finally 一定被执行,在 finally 子句中可以增加一些必须执行的语句。

异常语句捕捉和处理异常的机理是:当 try 子句中的代码产生异常时,按照 catch 子句的顺序查找异常类型。如果找到,则执行该 catch 子句中的异常处理语句。如果没有找到,则执行通用异常类型的 catch 子句中的异常处理语句。由于异常的处理是按照 catch 子句出现的顺序逐一检查 catch 子句,因此 catch 子句出现的顺序是很重要的。无论是否产生异常,一定执行 finally 子句中的语句。异常语句中不必一定包含所有 3 个子句,因此异常语句可以有以下 3 种可能的形式。

(1) try-catch 语句,可以有多个 catch 语句。

(2) try-finally 语句。

(3) try-catch-finally 语句,可以有多个 catch 语句。



本章小结

本章主要介绍了 C# 的应用方法,对.NET 开发环境、面向对象技术、C# 数据类型、C# 运算符以及 C# 控制语句进行了详细的介绍。

Visual Studio 是一个由基于组件的软件开发工具,内置了多种提高工作效率的功能,Microsoft.NET(以下简称.NET)框架是微软提出的新一代 Web 软件开发模型,.NET 已包含非常大的代码库,可以在通过面向对象语言,如 C# 等编程技术中来使用这些代码,这样可以极大提高程序设计人员的工作效率。

面向对象程序设计(OOP)是一种程序设计范型,同时也是一种程序开发的方法。已经被证实的是,面向对象程序设计推广了程序的灵活性和可维护性,并且在大型项目设计中广为应用。此外,支持者声称面向对象程序设计要比以往的做法更加便于学习,因为它能够让人们更简单地设计并维护程序,使得程序更加便于分析、设计、理解。

C# 是一种新的、面向对象的编程语言,是.NET 框架中新一代的开发工具,用 C# 编写的应用程序可以充分利用.NET 的框架体系带来的优点,既可以用来编写基于通用网络协议的 Internet 服务软件,也可以编写各种数据库、网络服务应用程序和 Windows 窗口界面程序。它简化了 C++语言在类、命名空间、方法重载和异常处理等方面的操作,使用组件辅助编程,十分容易掌握。



思考与练习

1. 根据本章的介绍编写一个记事本的程序。
2. 了解面向对象的技术,并将其应用到 C# 程序开发中。

第 3 章

ASP.NET

ASP.NET 是微软公司.NET 框架的一部分，是一种使嵌入网页中的脚本在因特网服务器执行的服务器端脚本技术，它可以在通过 HTTP 请求文档时再在 Web 服务器上动态创建。ASP.NET 是 ASP 技术的扩展，作为最新的动态网页开发技术，已经广泛应用于很多领域。

3.1 Web 窗体

3.1.1 Web 窗体概述

Web 窗体也称为 Web Form，在 ASP.NET 中是指用 ASP.NET 开发的网页。Web 窗体主要用来根据特定的信息生成动态的页面。Web 窗体提供了窗体设计器、编辑器、控件和调试功能，这些功能结合在一起，能够为浏览器和 Web 客户端设备快速生成基于服务器的可编程用户界面，极大地提高了开发效率。

1. Web 窗体结构

Web 窗体由 HTML 标记、控件和逻辑代码共同组成。当用户请求包含这些控件的网页时，被请求网页将首先在服务器端执行，以生成 HTML 文档，并回送客户端，浏览器再将结果显示给用户。

Web 窗体由可视化组件和用户接口逻辑组成。其中，前者指包含 HTML 标记及控件声明的部分(也就是可在浏览器上看到的部分)；后者则指用于实现服务器和用户交互的代码。在 Visual Studio.NET 开发环境中添加 Web 窗体，默认可视化组件与用户接口逻辑分处不同的文件中。也可将上述两个部分共处同一文件中，只需在添加 Web 窗体时，取消将代码放在单独的文件中即可。Web 窗体结构如图 3-1 所示。

2. Web 窗体的生命周期

一般来讲，Web 窗体页的生命周期类似于在服务器上运行的任何 Web 进程的生命周期。具体过程如下。

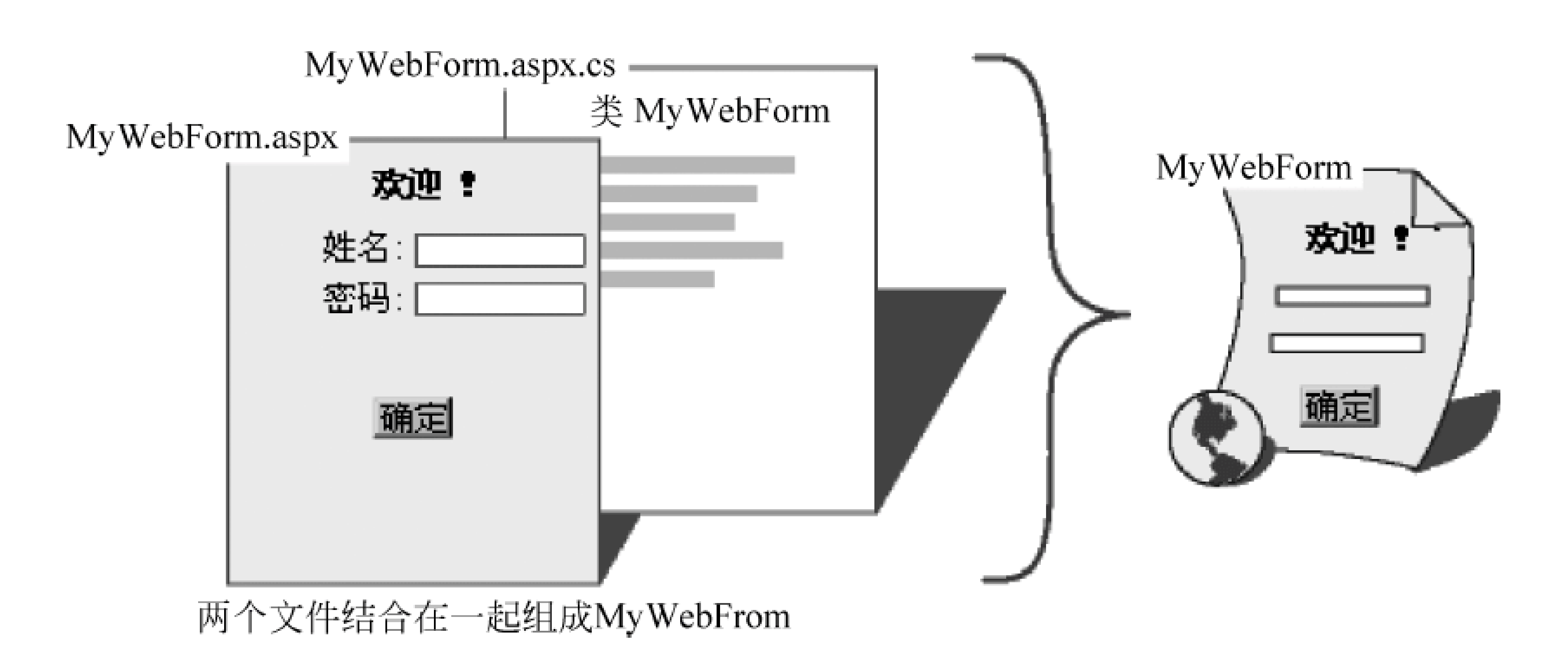


图 3-1　Web 窗体结构

(1) 用户通过浏览器请求页面。

(2) 加载并初始化页面和控件。

(3) 如果请求的页面是回发的结果，控件状态从视图状态加载，并应用用户提交的任何改变。

(4) 页面事件处理程序和用于用户操作触发事件的事件处理程序被执行。

(5) 将控件状态保存到视图状态中。

(6) 将页面的 HTML 输出到浏览器中。

(7) 卸载页面和控件。

3.1.2　添加 Web 窗体

创建网站后，用户可根据需要在网站中添加 Web 窗体。操作步骤如下。

(1) 在"解决方案资源管理器"中右击相应的网站，在弹出的快捷菜单中选择"添加"→"添加新项"→"Web 窗体"命令，弹出图 3-2 所示的添加 Web 窗体对话框。



图 3-2　添加 Web 窗体对话框

（2）在此对话框中，选择语言并输入 Web 窗体名称，选择是否将代码放在单独的文件中后，单击“添加”按钮，即可将新的 Web 窗体添加到网站中。

3.1.3 Web 窗体的默认代码

1. Web 窗体的默认源代码

```
<%@ Page Language = "C#" AutoEventWireup = "true" CodeFile = "Default.aspx.cs" Inherits =
"_Default" %><!DOCTYPE html>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head runat = "server">
<meta http-equiv = "Content-Type" content = "text/html; charset = utf-8"/>
<title></title>
</head><body>
<form id = "form1" runat = "server"><div>
//放窗体控件
</div></form></body></html>
```

说明如下。

（1）＜%...%＞语句块用于存放在服务器端执行的代码及指令。

（2）＜@Page＞标记表示“页面属性”，定义了 Web 页面特定的属性。一个.aspx 文件只能包含一个＜@Page＞标记。＜@Page＞标记的常用属性如表 3-1 所示。

表 3-1 ＜@Page＞标记的常用属性

属　性	用　途
AutoEventWireup	决定是否自动装载事件(如 Page_Load)的处理器。默认值为 True
Buffer	决定显示输出是在被发送到客户端之前进行缓存，还是直接发送显示。默认值为 True
CodeFile	为页面指定代码隐藏类的文件名
Inherits	为页面指定一个代码隐藏类
Language	指明用于编译该页的语言编译器

（3）＜Form＞标记具有以下属性。

① Mothod 属性。该属性定义了将控件的值送回服务器端的传输方式，可以设置的值有 Post 和 Get。

② Runat 属性。如果该属性的值被设置为 server，能使窗体将控件信息送回到服务器端的 ASP.NET 页面进行处理；否则该窗体作为一般的 HTML 窗体运行。

2. Web 窗体的默认程序代码

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
public partial class _Default : System.Web.UI.Page
{   protected void Page_Load(object sender, EventArgs e)
    {
        //此处放置页面加载时代码
    }
}
```

说明如下。

(1) 命名空间是一种逻辑命名方案，是类的集合，用 using 指令导入。

(2) using 指令用于导入命名空间。可将命名空间导入 ASP.NET 应用程序文件中。一个指令只能导入一个命名空间，如果要导入多个命名空间，应使用多个 using 指令来执行。大部分常用的命名空间在默认状态下都被自动导入，不需要手动导入。

3.2　ASP.NET 服务器控件

ASP.NET 提供了大量可在 ASP.NET 网页上选用的 Web 服务器控件，熟练使用这些控件有助于提高 Web 应用程序的开发效率。

本节主要学习用于创建窗体的常用 Web 服务器控件。通过本节知识的学习，读者首先对服务器控件有些初步了解。掌握 TextBox、Label、Button、DropDownList 等常用控件的使用方法，并利用本节知识设计和实现用户注册页面。

3.2.1　服务器控件概述

ASP.NET 服务器控件是运行在服务器端并且封装了用户界面和其他功能的组件。控件的含义表明它不仅是具有呈现外观作用的元素，而且是一种对象，一种定义 Web 应用程序用户界面的组件。

(1) ASP.NET 提供了与 HTML 控件相对应的基本 Web 服务器控件。

(2) Web 服务器控件类都包含在 System. Web. UI. WebControls 命名空间下面。

(3) 在 ASP.NET 中，Web 服务器控件是使用相应的标记来编写控件的。

1. Web 服务器控件工作原理

Web 服务器控件工作原理如图 3-3 所示。

在网站的页面中添加控件有两种方法。

(1) 在工具箱中双击控件，则控件以默认位置、默认风格直接插入页面中。

(2) 将工具箱中的控件直接拖动到页面指定位置。

2. 服务器控件的属性和事件

1) 服务器控件的属性

服务器控件的属性是指控件中具有的与用户界面特征相关的字段或与运行状态有关的字段。服务器控件属性的使用方式有 3 种。

(1) 在与控件对应的属性窗口中设置属性。“属性”窗口用来设置控件的属性，当页面初始化时，控件的这些属性将被应用到控件。当开发人员选择了相应的属性后，属性栏

中会简单地介绍该属性的作用，如图 3-4 所示。



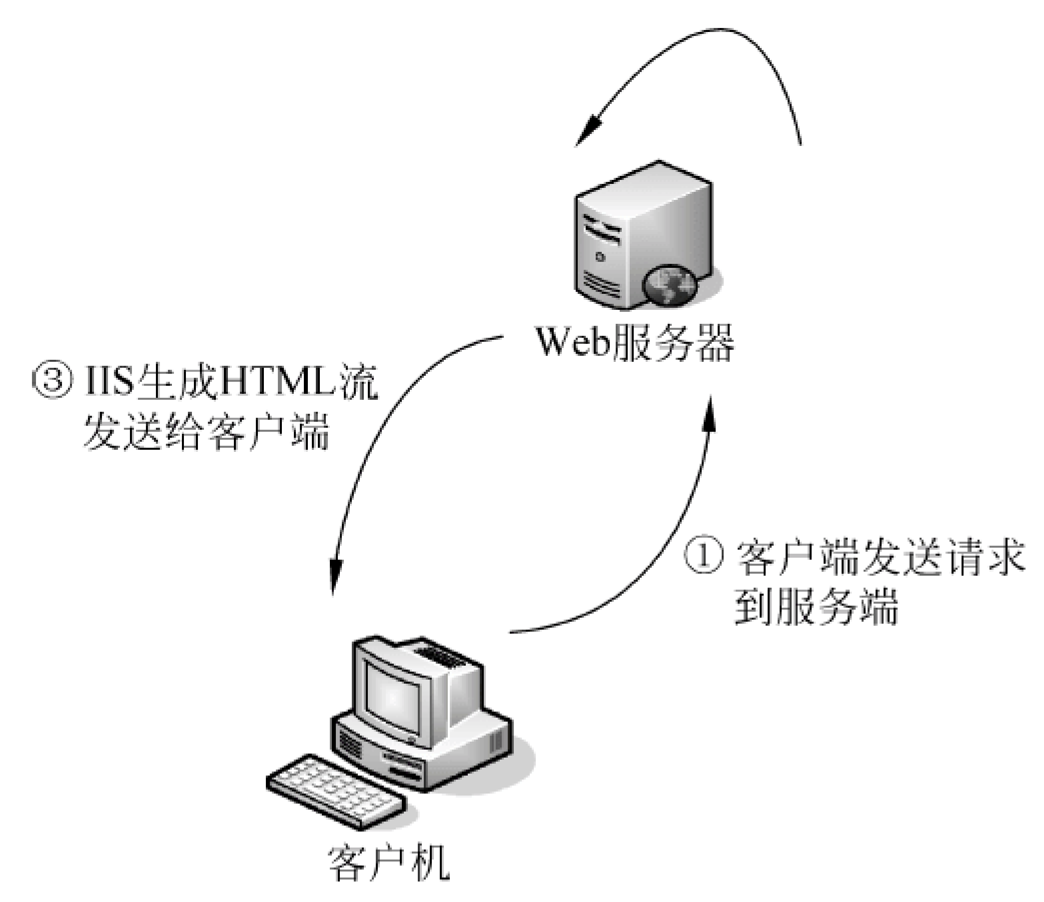



图 3-3 Web 服务器控件工作原理



图 3-4 “属性”窗口

(2) 在定义控件的标记里设置属性。以标记<asp:控件名…>开始，控件中包含 runat="server"属性，表示为服务器控件；包含 ID 属性，用于标识控件。

在源代码中可使用下列两种语法对控件进行声明。示例代码如下。

```
<asp:Label ID = "Label1" runat = "server" Text = "Label"></asp:Label>
```

或

```
<asp:Label ID = "Label1" runat = "server" Text = "Label"/>
```

(3) 在程序代码中设置控件属性。示例代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{   Label1.Visible = false ;                    //在 Page_Load 中设置 Label1 的可见性}
```

2) 服务器控件的事件

事件是指程序得以运行的触发器(如 Button 控件的 Click 事件等)，当用户与 Web 页面进行交互时被触发，并通过执行事件程序做出相应的响应。

(1) Web 控件的事件工作方式与传统的 HTML 标记的客户端事件工作方式有所不同，这是因为 HTML 标记的客户端事件是在客户端引发和处理的，而 ASP.NET 页面中 Web 控件的事件是在客户端引发，在服务器端处理。

(2) 所有的 Web 事件都包括两个参数：第 1 个参数表示引发事件的对象；第 2 个参数表示包含该事件特定信息的事件对象。

例如，普通按钮的单击事件，代码如下：

```
protected void Button1_Click(object sender, EventArgs e)
{   //在此处添加单击事件处理代码 }
```

进入代码编辑窗口有两种方法。

(1) 双击控件,即进入控件编程界面。

(2) 在属性面板上部单击事件切换图标,选定特定事件后双击相应事件,即可启动代码编辑窗口。

3.2.2 服务器控件的共有属性

1. 外观属性

(1) BackColor。BackColor 属性用于设置对象的背景色,其属性设定为颜色名称或是#RRGGBB 的格式。

(2) ForeColor 属性。ForeColor 属性用于设置对象的前景色,其属性设定为颜色名称或#RRGGBB 的格式。

(3) Border 属性。BorderWidth 属性可以用于设定控件的边框宽度,单位是像素;BorderColor 属性用于设定边框的颜色,其属性的设定值为颜色名称或是#RRGGBB 的格式;BorderStyle 属性用来设定对象的边框样式。

(4) Font 属性。Font 属性有以下几个子属性,分别表现不同的字体特性。

① Font-Bold:如果属性值设定为 True,则会变成粗体显示。

② Font-Italic:如果属性值设定为 True,则会变成斜体显示。

③ Font-Name:设置字体的名字。

④ Font-Size:设置字体大小,共有 9 种大小可供选择,即 Smaller、Larger、XX-Small、X-Small、Small、Medium、Large、X-Large 或者 XX-Large。

⑤ Font-Strikeout:如果属性值设定为 True,则文字中间显示一条删除线。

⑥ Font-Underline:如果属性值设定为 True,则文字下面显示一条底线。

2. 行为属性

(1) Enabled 属性。Enabled 属性用于设置禁止控件还是使能控件。当该属性值为 False 时,控件为禁止状态;当该属性值为 True 时,控件为使能状态。对于有输入焦点的控件,用户可以对控件执行一定的操作。

(2) ToolTip 属性。ToolTip 属性用于设置控件的提示信息。在设置了该属性值后,当鼠标停留在 Web 控件上一小段时间后就会出现 ToolTip 属性中设置的文字。

(3) Visible 属性。Visible 属性决定了控件是否被显示,如属性值为 True 将显示该控件;否则隐藏该控件。

3. 可访问性

(1) AccessKey 属性。AccessKey 属性用来为控件指定键盘的快捷键,这个属性的内容为数字或英文字母。例如,设置为 A,那么使用控件时用户按下 Alt+A 组合键就会自动将焦点移动到这个控件的上面。

(2) TabIndex 属性。TabIndex 属性用来设置 Tab 键的顺序。当用户按下 Tab 键时,输入焦点将从当前控件跳转到下一个可以获得焦点的控件,TabIndex 就是用于定义这种跳转顺序的。

4. 布局属性

Height 和 Width 属性用于设置控件的高度和宽度，单位是 px(像素)。

3.2.3 常用服务器控件

1. 标签控件(Label)

在 Web 应用中，希望显示的文本不能被用户更改，或者当触发事件时，某一段文本能够在运行时更改，则可以使用标签控件 Label。

Label 控件常用的属性有 ID、Text 和 Font 属性等。其中，ID 表示控件标识；Text 表示控件显示的文本内容；Font 表示字体格式设置，如大小、颜色等。

将标签控件拖放到页面后，将自动生成标签控件的声明代码，示例代码如下：

```
<asp:Label ID = "Label1" runat = "server" Text = "Label"></asp:Label>
```

同样，标签控件的属性可在相应的程序代码中进行设置，示例代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{ Label1.Text = "欢迎!"; //标签赋值}
```

上述代码在页面载入时将 Label1 的文本属性设置为"欢迎!"。

2. 文本框控件(TextBox)

在 Web 开发中，Web 应用程序通常需要和用户进行交互，如用户注册、登录等，就需要文本框控件来接受用户输入的信息。文本框控件也可以设置为只读，用于文本显示。

1) 文本框控件的属性

文本框控件常用属性如表 3-2 所示。

表 3-2 文本框控件常用属性

属　性	说　明
ID	控件唯一标识
Text	控件要显示的文本
TextMode	控件的输入模式，有 SingleLine(单行)、MultiLine(多行)和 Password(密码) 3 种，默认为 SingleLine
Width	控件的宽度
MaxLength	控件可接收的最大字符数
AutoPostBack	控件内容修改后，是否自动发回到服务器。如果将控件的该属性设置为 True，则控件内容修改后会使页面自动发回到服务器
Visible	控件是否可见
Enabled	控件是否可用
ReadOnly	控件是否只读，如果将此属性设置为 True，则文本框内的值是无法被修改的
Rows	控件中文本显示的行数，该属性在 TextMode 为 MultiLine 时有效

2) 常见代码格式

```
<asp:textbox id = "控件名" runat = "server" text = "文本内容"></asp:textbox>
```

或

```
<asp:textbox id="控件名" runat="server" text="文本内容" />
```

【例 3-1】 利用 TextBox 控件制作用户登录页面，如图 3-5 所示。要求密码输入时以黑点显示。

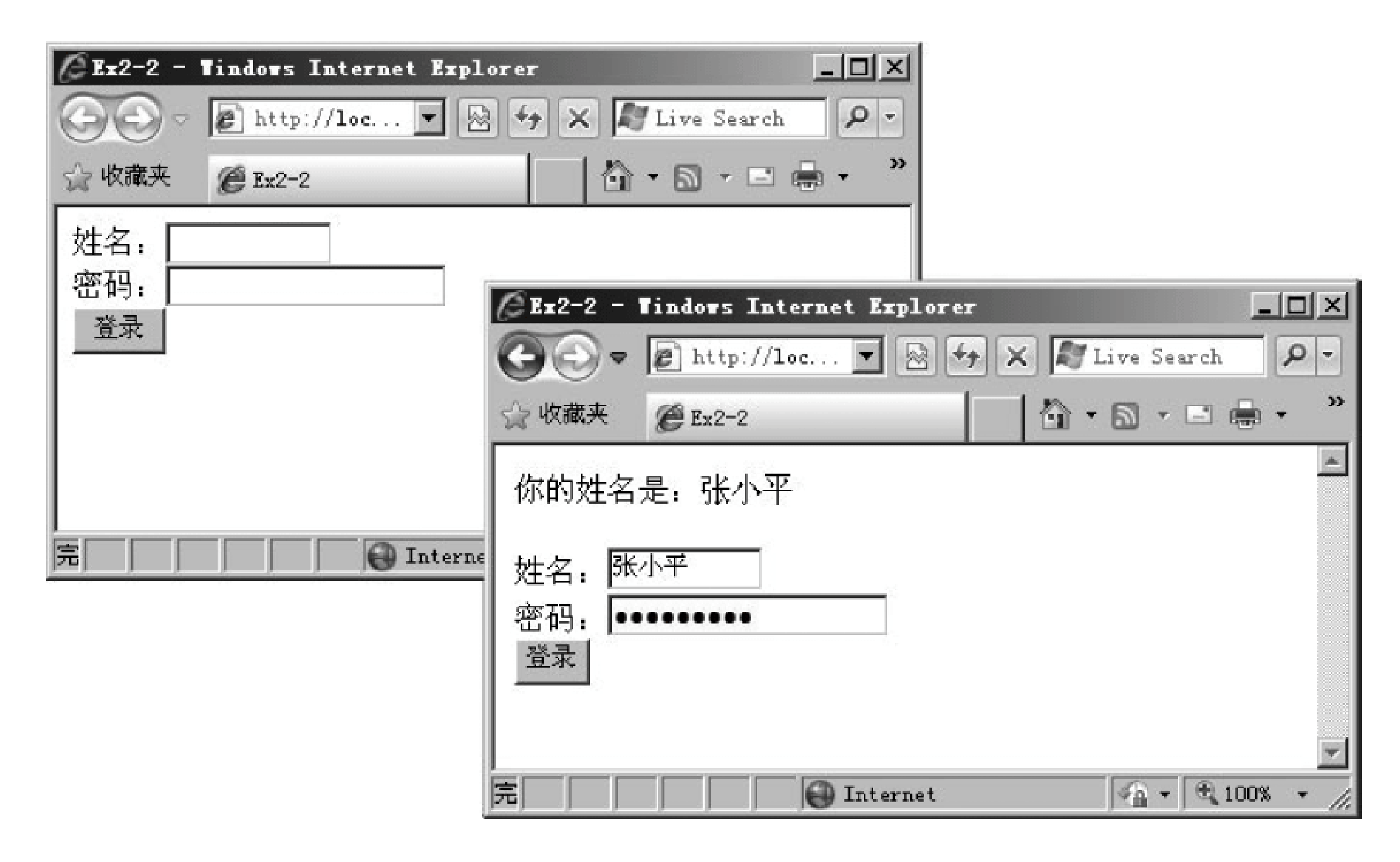


图 3-5　利用 TextBox 控件制作用户登录页面

主要代码如下。

```
<div>姓 名: <asp:TextBox ID="txtname" runat="server" AutoPostBack="True" Width=
"80px" MaxLength="3" ontextchanged="txtname_TextChanged" />
     密码: <asp:TextBox ID="TextBox2" runat="server" TextMode="Password" />
     <asp:Button ID="Button1" runat="server" Text="登录" />
     </div>
     protected void txtname_TextChanged(object sender, EventArgs e)
     {Response.Write("你的姓名是: " + txtname.Text);}
```

3. 超链接控件(HyperLink)

HyperLink 控件用于建立文本超链接或图片超链接。

可使用下列两种语法进行声明。

```
<ASP:HyperLink Runat="Server" Id="…" NavigateUrl="…" Text="…" ImageUrl="…"
Target="…" />
```

或

```
<ASP:HyperLink Runat="Server" Id="…" NavigateUrl="…" ImageUrl="…"
Target="…">超链接文本</ASP:HyperLink>
```

说明如下。

- NavigateUrl：给出或设置所链接文档的 URL。

- ImageUrl：给出或设置超链接图片。
- Text：给出或设置超链接文本。
- Target：给出或设置目标框架（或窗口）名称。例如，将 Target 设为"_blank"，可将所链接的文档显示在新窗口中。

4. 图片控件(Image)

Image 控件用于在网页中插入图片。其声明语法如下。

```
<ASP:Image Runat = "Server" Id = "..." ImageUrl = "..." AlternateText = "..." ... />
```

说明如下。

- ImageUrl 属性：用于给出或设置图片位置。
- AlternateText 属性：用于给出或设置替换文本。

5. 单选按钮控件(RadioButton)

RadioButton 控件是单选按钮控件，当用户选择某个单选按钮时，同组中的其他选项不能被同时选中。常用属性如表 3-3 所示。

表 3-3　单选按钮控件常用属性

属　　性	说　　明
ID	控件唯一标识
Text	控件关联的文本标签
GroupName	控件所属的控件组名
Checked	控件是否被选中
AutoPostBack	单击控件时是否自动发回到服务器
Enabled	判断控件是否可用

【例 3-2】 利用 RadioButton 控件实现考试系统中单选题的操作，如图 3-6 所示。用户不选择答案单击“提交”按钮时，页面弹出“请选择答案!”提示；当用户选择正确答案 B 时，页面提示“恭喜你，回答正确!”；否则提示“对不起，正确答案是 B!”。



图 3-6　考试系统中的单选题

主要代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
  if (R1.Checked == false && R2.Checked == false && R3.Checked == false)
  Response.Write("<script>alert('请选择答案!')</script>");
  else
  {
    if (R2.Checked == true)
    Response.Write("<script>alert('恭喜你,回答正确!')</script>");
    else
    Response.Write("<script>alert('对不起,正确答案是 B!')</script>");
  }
}
```

6. 单选按钮组控件(RadioButtonList)

单选按钮组控件 RadioButtonList 有效地解决了每个 RadioButton 控件在 RadioButton 组中相互独立的问题,为读者提供了一组 RadioButton,大大方便了用户操作。常用属性如表 3-4 所示。

表 3-4 单选按钮组控件常用属性

属　性	说　明
AutoPostBack	单击控件时是否自动发回到服务器,响应 OnSelectedIndexChanged 事件
CellPading	各项目之间的距离,单位是像素
Items	返回 RadioButtonList 控件中 ListItem 的对象
RepeatDirection	选择项目的排列方向,默认为 Vertical
SelectedItem	返回被选择的 ListItem 对象
TextAlign	设置各项目所显示文字在按钮左边还是右边,默认为 Right

【例 3-3】 利用 RadioButtonList 控件实现性别单选和提示文字即时更新,如图 3-7 所示。如用户选择“男”,页面下面的文字立即改变为“你选择的是:男;对应的值为:1”。



图 3-7 利用 RadioButtonList 控件实现性别单选

主要代码如下。

```
protected void rblsex_SelectedIndexChanged(object sender, EventArgs e)
```

```
{   lbl1.Text = rblsex.SelectedItem.Text;
    lbl2.Text = rblsex.SelectedItem.Value;
}
```

7. 多选按钮控件(CheckBox)

CheckBox 控件和 RadioButton 控件的区别在于前者允许多选,后者只能单选。CheckBox 控件和 RadioButton 控件的常用属性基本相同。

【例 3-4】 利用 CheckBox 控件实现考试系统中不定项选择题的操作,如图 3-8 所示。当用户选择 A、B、C 这 3 个答案时,提示"回答正确";否则提示"错误,正确答案为 ABC"。



图 3-8 利用 CheckBox 控件实现考试系统中不定项选择题

主要代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (!ckb1.Checked && !ckb2.Checked && !ckb3.Checked && !ckb4.Checked)
        Response.Write("<script>alert('请选择答案!')</script>");
    else
    {   if (ckb1.Checked && ckb2.Checked && ckb3.Checked)
            Response.Write("<script>alert('回答正确')</script>");
        else
            Response.Write("<script>alert('错误,正确答案为 ABC')</script>");
    }
}
```

8. 多选按钮组控件(CheckBoxList)

用 CheckBox 控件可以实现多选功能,但在判断被选中的选项时需要对每一个对象都进行判断。CheckBoxList 控件和 RadioButtonList 控件类似,可以方便地判断用户选中的选项。

【例 3-5】 利用 CheckBoxList 控件实现去过的城市选择功能页面,如图 3-9 所示。根据用户选择的不同,页面下方出现的选择结果动态变化。

主要代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
```

```
    string result = "";
    for (int i = 0; i < ckl1.Items.Count; i++)
    {    if (ckl1.Items[i].Selected)
          result += ckl1.Items[i].Text + " ";
    }
    if  (result == "")
    lblmes.Text = "您都没去过。";
    else
    lblmes.Text = "您去过的城市是：" + result;
}
```



图 3-9 利用 CheckBoxList 控件实现去过的城市选择

9. 下拉列表控件(DropDownList)

DropDownList 控件是一个下拉式列表控件，用户可从下拉式列表中选择单一选项。

【例 3-6】 利用 DropDownList 控件实现用户出生地选择功能，如图 3-10 所示。根据用户所选择的出生地不同，单击“提交”按钮后提示不同的信息。



图 3-10 利用 DropDownList 控件实现用户出生地选择

主要代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{    lblmes.Text += ddlbir.SelectedItem.Text;}
```

10. 日历控件(Calendar)

Calendar 控件是日历控件，用于选择日期。可以结合 TextBox 控件一起使用，实现

日期输入,从而规范并简化日期格式输入。

【例 3-7】 利用 Calendar 控件实现用户入团日期输入功能,如图 3-11 所示。用户通过选择 Calendar 控件的日期,使其自动出现在 TextBox 控件文本框中。

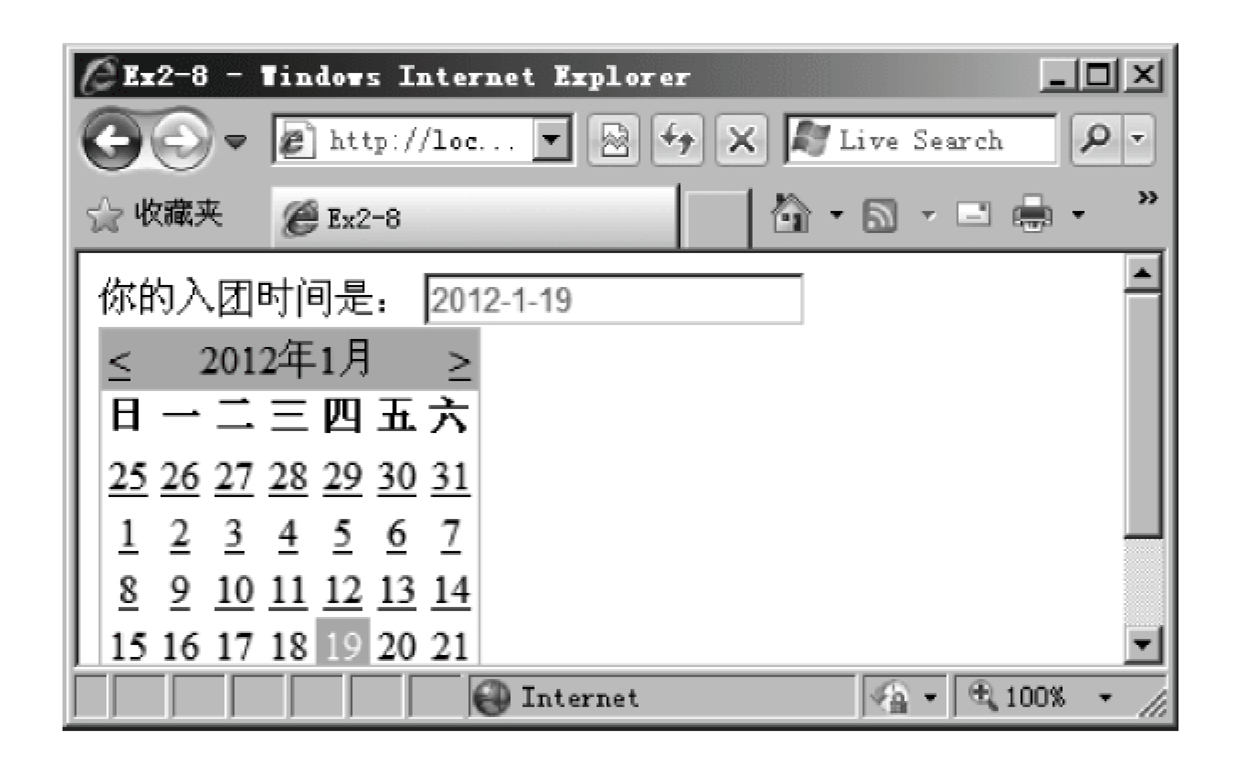


图 3-11 利用 Calendar 控件实现用户入团日期输入

主要代码如下。

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{    txtdate.Text = Calendar1.SelectedDate.ToShortDateString();}
```

11. 按钮控件(Button、LinkButton、ImageButton)

在 Web 应用程序和用户交互时,按钮控件是非常必要的。按钮控件能够触发事件,或者将网页中的信息回传给服务器。

在 ASP.NET 中,包含三类按钮控件:普通按钮、超链接按钮、图片按钮,分别为 Button、LinkButton、ImageButton。

(1) 按钮控件的常用通用属性如下。

① Causes Validation:按钮是否导致激发验证检查。

② CommandArgument:与此按钮关联的命令参数。

③ CommandName:与此按钮关联的命令。

(2) 按钮声明语句如下。

下面的语句分别声明了普通按钮、超链接按钮、图片按钮,示例代码如下。

```
<asp:Button ID = "Button1" runat = "server" Text = "确定" />
<asp:LinkButton ID = "LinkButton1" runat = "server">超链接按钮</asp:LinkButton>
<asp:ImageButton ID = "ImageButton1" runat = "server" ImageUrl = "~/images/date.GIF" />
```

(3) 按钮的单击事件 Click。

示例代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{ Label1.Text = "普通按钮被触发";   }
```

(4) 按钮的 Command 命令事件。

按钮控件中,Click 事件并不能传递参数,所以处理的事件相对简单。而 Command 事

件可以传递参数，负责传递参数的是按钮控件的 CommandArgument 和 CommandName 属性。

【例 3-8】 将 CommandArgument 和 CommandName 属性分别设置为 Hello! 和 Show，单击创建一个 Command 事件并在事件中编写相应代码，如图 3-12 所示。

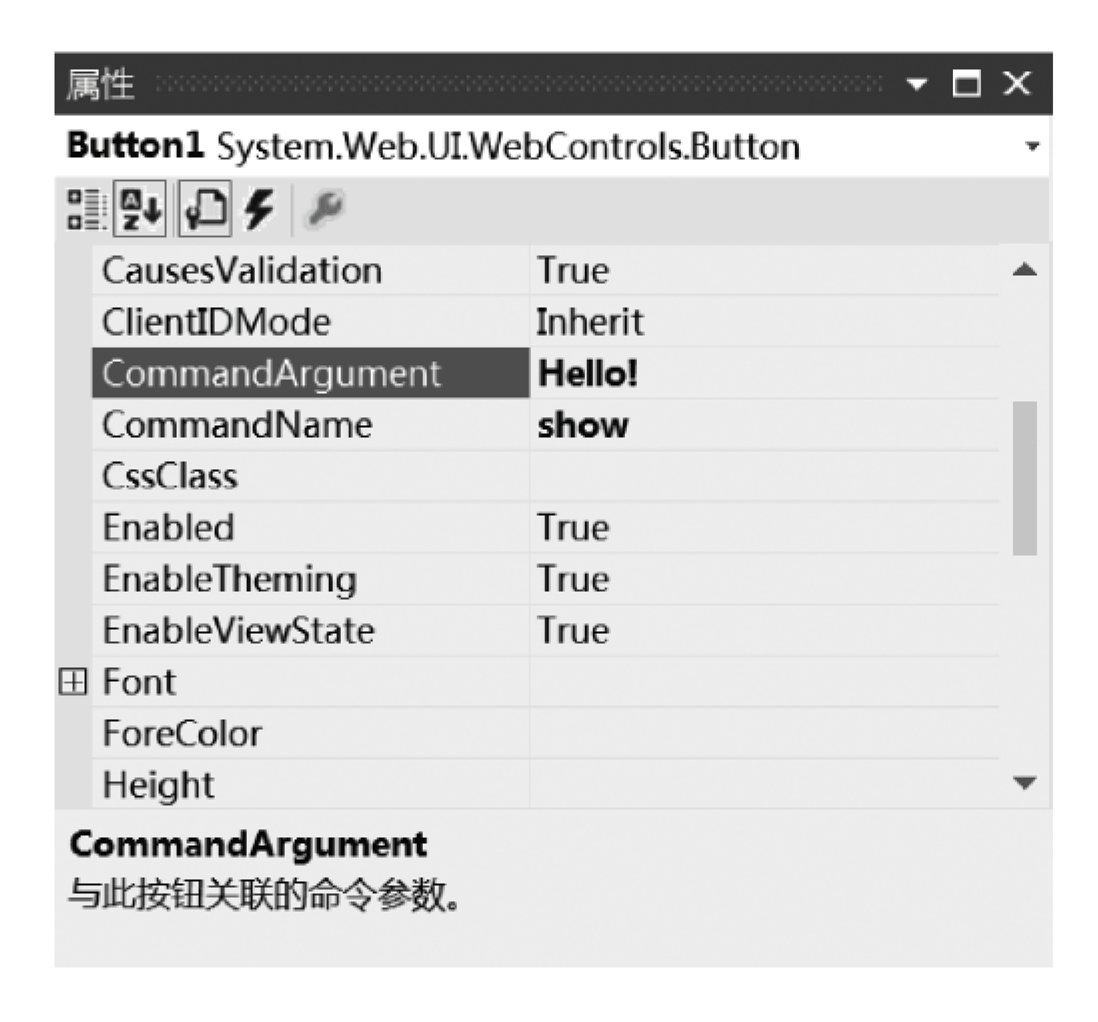


图 3-12 CommandArgument 和 CommandName 属性

主要代码如下。

```
protected void Button1_Command(object sender, CommandEventArgs e)
{
  if (e.CommandName == "Show")
  {
    Label1.Text = e.CommandArgument.ToString();
  }
}
```

1) Button 控件

Button 控件是读者使用频率最高的控件之一，用户通过单击 Button 来执行该控件的 Click 事件。Button 控件的常用属性有 Id、Text 及 onclick 事件。

【例 3-9】 利用 Button 控件控制 Calendar 控件的显示，效果如图 3-13 所示。单击“显示日历”按钮，出现日历；用户选择日期后，日期出现在文本框中，日历窗口关闭。

主要代码如下。

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{    txtdate.Text = Calendar1.SelectedDate.ToShortDateString();
     Calendar1.Visible = false ;}
protected void Button1_Click(object sender, EventArgs e)
{     Calendar1.Visible = true;}
```

2) LinkButton 控件

LinkButton 控件又称为超链接按钮控件，该控件在功能上与 Button 控件相同，但样



图 3-13　利用 Button 控件控制 Calendar 控件的显示

式以超链接形式显示。LinkButton 控件有一个 PostBackUrl 属性，该属性用于设置单击控件时链接到的网址。

【例 3-10】　利用 LinkButton 控件 PostBackUrl 属性实现超链接功能，如图 3-14 所示。用户单击“打开 Ex2-9”链接，将转向 Ex2-9. aspx 页面。

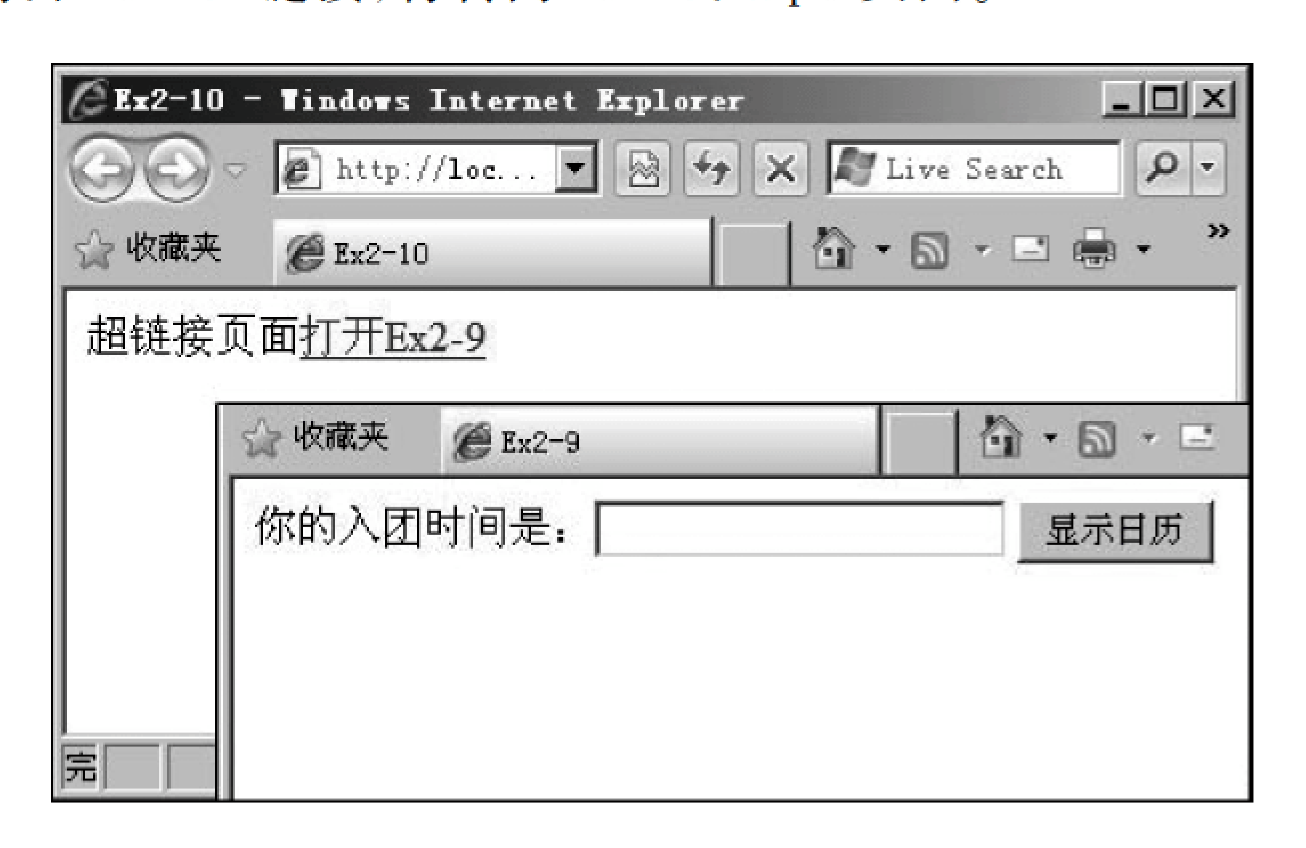


图 3-14　利用 LinkButton 控件 PostBackUrl 属性实现超链接

主要代码如下。

```
<form id = "form1" runat = "server">
<div>超链接页面<asp:LinkButton ID = "LinkButton1" runat = "server" PostBackUrl = "～/Ex2
-9.aspx">打开 Ex2-9</asp:LinkButton>
</div>
</form>
```

3）ImageButton 控件

ImageButton 控件是图片按钮控件，用户单击控件上的图片引发控件的 Click 事件。ImageButton 控件有一个 ImageUrl 属性，该属性用于设置按钮上显示的图片位置。

【例 3-11】　利用 ImageButton 控件达到美化按钮的效果，如图 3-15 所示。

主要代码如下。



图 3-15　利用 ImageButton 控件美化按钮

```
<div>
你的入团时间是:
<asp:TextBox ID="txtdate" runat="server" Enabled="False"/>
<asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="~/images/date.GIF"
            onclick="ImageButton1_Click" />
<asp:Calendar ID="Calendar1" runat="server" Visible="False"></asp:Calendar>
</div>
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{ Calendar1.Visible = true;}
```

12. 文件上传控件(FileUpload)

FileUpload 控件是用于客户端文件上传到服务器的控件。该控件显示一个文本框和一个浏览按钮,用户可以通过“浏览”按钮选择文件。FileUpload 控件有一个 Save As 方法,用于将上传的文件保存到服务器。文件上传控件常用属性如表 3-5 所示。

表 3-5　文件上传控件常用属性

属　　性	说　　明
ID	控件唯一标识
FileName	获取上传文件在客户端的文件名称
HasFile	获取一个布尔值,用于表示控件是否已经包含一个文件
PostedFile	获取一个与上传文件相关的 HttpPostedFile 对象,使用该对象可以获取上传文件的相关属性

【例 3-12】 利用 FileUpload 控件实现文件上传操作,如图 3-16 所示。用户单击页面“浏览...”按钮,选择要上传的文件,单击“上传”按钮,文件将上传到服务器网站的根目录下。

主要代码如下。

```
<div>
请选择上传的文件: <asp:FileUpload ID="fulfile" runat="server" />
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="上传" />
<asp:Label ID="lblmes" runat="server" Text=""></asp:Label>
```



图 3-16　利用 FileUpload 控件实现文件上传

```
</div>
protected void Button1_Click(object sender, EventArgs e)
{
  if (fulfile.HasFile)
  {
    string strname = fulfile.FileName;
    fulfile.SaveAs(Server.MapPath(strname));
    lblmes.Text += "文件：" + strname + "已上传到了根目录!";
  }
  else
  Response.Write("请选择上传文件!");
}
```

13. 表格控件(Table)

网站开发过程中，表格是页面布局的一种重要手段。使用 Table 表格、tr 表格行和 td 表格单元格进行页面布局，操作简单、快捷，大大提高了开发效率。表格控件常用属性如表 3-6 所示。

表 3-6　表格控件常用属性

属　性	说　明
Boder	表格边框宽度
CellPadding	单元格边框与内容的间距
CellSpacing	单元格间距
Align	表格、单元格水平方向对齐方式，有 Left、Right 和 Center 3 种
Valign	单元格竖直方向对齐方式，有 Baseline、Bottom、Middle 和 Top 4 种
Style	表格、单元格样式

【例 3-13】 利用表格控件实现系统登录页面布局设计，如图 3-17 所示。

14. 验证控件

验证控件在服务器代码中执行输入检查。当用户向服务器提交页面之后，服务器将



图 3-17 利用表格控件实现登录页面布局设计

逐个调用验证控件来检查用户输入。如果在任意输入控件中检测到验证错误，则该页面将自行设置为无效状态，以便在代码运行之前测试其有效性。ASP.NET 提供了 5 种验证控件和一个验证结果信息汇总控件，验证控件的类型和作用如表 3-7 所示。

表 3-7 验证控件的类型和作用

验证类型	控件名称	作用
必需项	RequiredFieldValidator	验证某个控件的内容是否被改变
与某值的比较	CompareValidator	将用户的输入与一个常数值或者另一个控件或特定数据类型的值进行比较（使用小于、等于或大于比较运算符）
范围检查	RangeValidator	验证某个值是否在要求的范围内
模式匹配	RegularExpressionValidator	验证相关输入控件的值是否匹配正则表达式指定的模式。此类验证使用户能够检查可预知的字符序列，如电子邮件地址、电话号码、邮政编码等内容中的字符序列
用户自定义	CustomValidator	使用用户自己编写的验证逻辑检查用户输入。此类验证使用户能够检查在运行时派生的值
显示验证信息	ValidationSummary	显示来自页面上所有验证控件的错误信息。它只能与上述验证控件一起使用，不能单独执行验证

对于一个输入控件，可以附加多个验证控件。

1）RequiredFieldValidator 控件

RequiredFieldValidator 控件用于对一些必须输入的信息进行验证，如果对应的输入控件没有输入信息，则提示错误。该控件的属性主要有以下几个。

（1）ControlToValidate 属性：关联要被验证的控件。

（2）InitialValue 属性：用于获取或者设置要被检验的初始值。

（3）ErrorMessage 属性：验证不通过时显示的错误信息。

（4）Text 属性：控件中显示的信息。

【例 3-14】 对于用户登录页面,检查用户名和密码是否为空,如图 3-18 所示。

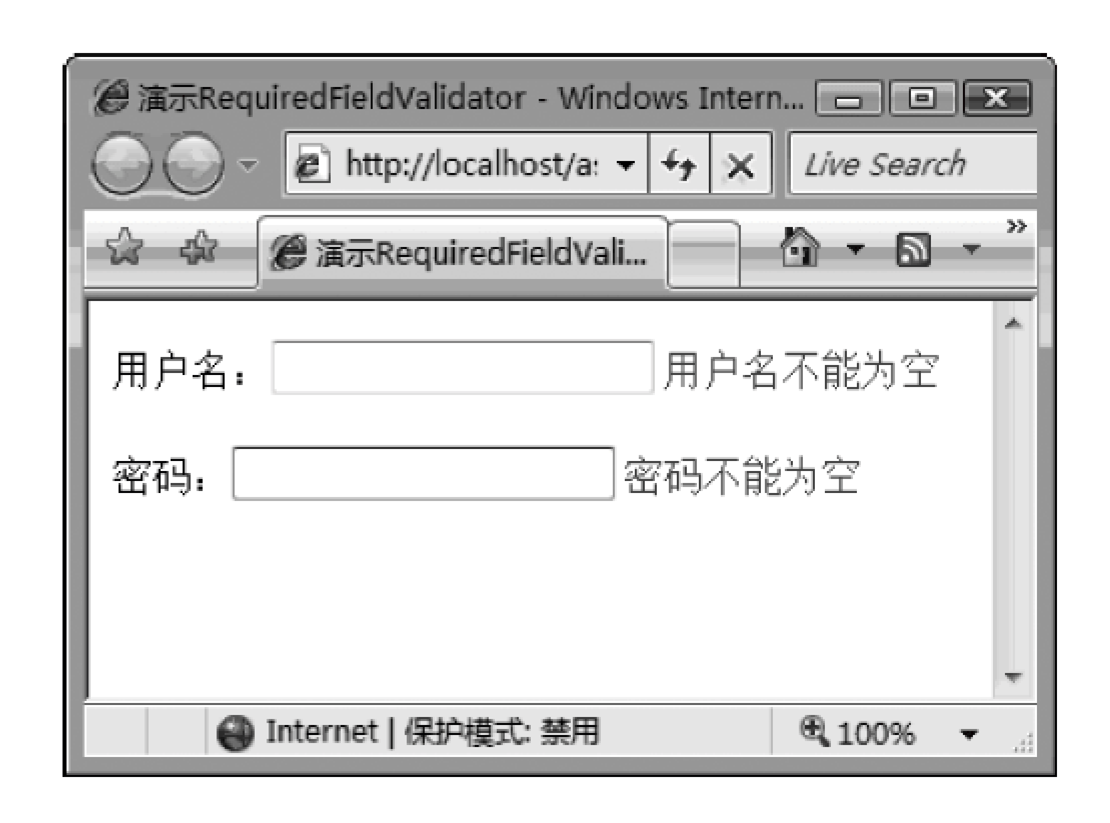


图 3-18 必需项验证

主要代码如下。

```
<asp:TextBox runat = "server" ID = "TextBox1" ></asp:TextBox>
<asp: RequiredFieldValidator ID = " RequiredFieldValidator1 " runat = " server "
ControlToValidate = "TextBox1" ErrorMessage = "用户名不能为空" ForeColor = "Red">
</asp:RequiredFieldValidator>
```

2) CompareValidator 控件

CompareValidator 控件用于将用户输入的值和其他控件的值或者常数进行比较。该控件的主要属性有以下几个。

(1) ControlToValidate 属性:指定要验证的输入控件。

(2) Type 属性:指定了两个比较值的数据类型。

(3) Operator 属性:指定了进行比较的类型。

【例 3-15】 使用 CompareValidator 控件,验证用户在第二个 TextBox 中输入的数字是否大于在第一个 TextBox 中输入的数字,如图 3-19 所示。



图 3-19 比较验证控件

主要代码如下。

```
最大值: <asp:TextBox runat = "server" ID = "TextBox1" TextMode = "SingleLine"></asp:TextBox>
```

```
最小值:<asp:TextBox runat="server" ID="TextBox2" TextMode="SingleLine"></asp:TextBox>
<asp: CompareValidatorid = " CompareValidator1 " runat = " server "  ForeColor = " Red"
ControlToValidate="TextBox1" ControlToCompare="TextBox2" Type="Double"
Operator="GreaterThanEqual" ErrorMessage="最大值不能小于最小值!">
</asp:CompareValidator>
```

3）RangeValidator 控件

RangeValidator 控件用于测试输入控件的值是否在指定范围内，直接继承于 BaseCompareValidator。该控件的主要属性有以下几个。

（1）MaximumValue：指定有效值的最大值。

（2）MinimumValue：指定有效值的最小值。

（3）ErrorMessage：指定验证失败时要显示的错误信息。

【例 3-16】 对用户在网页输入的年龄进行检查，如图 3-20 所示。

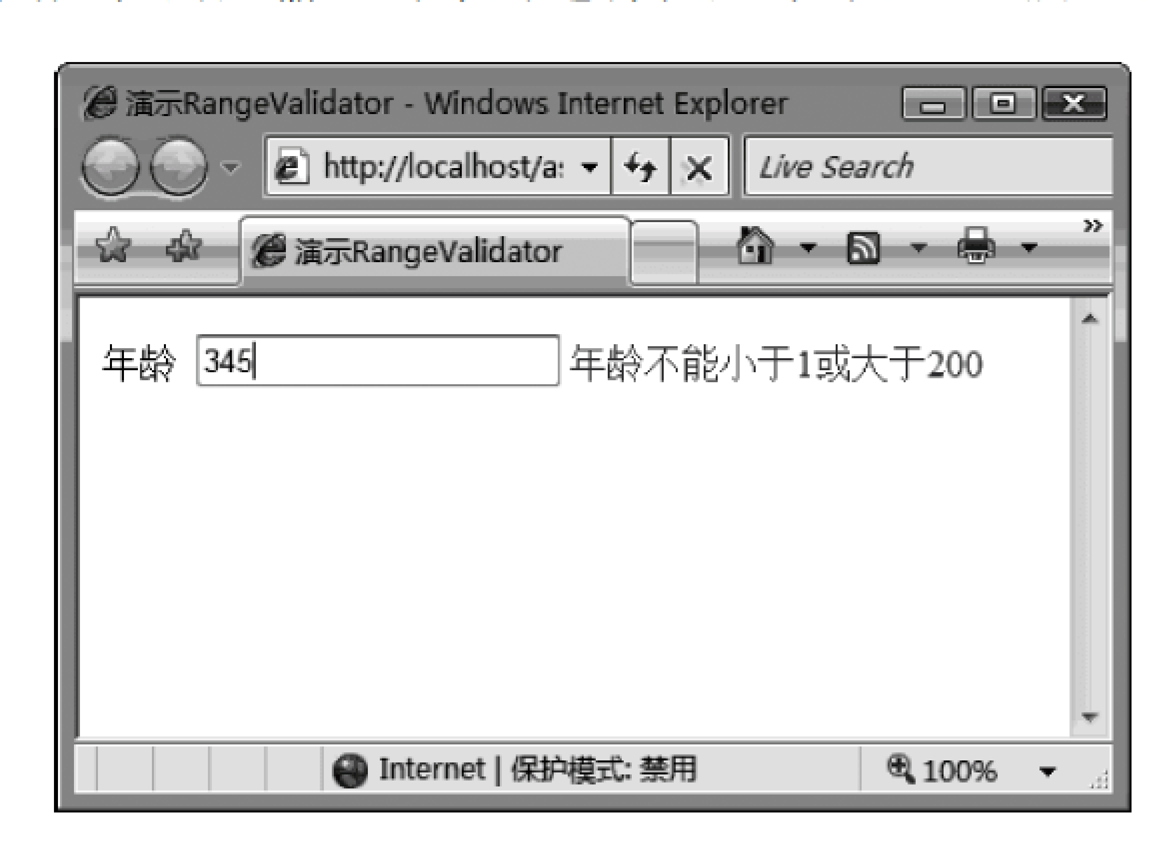


图 3-20 范围验证控件

主要代码如下。

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RangeValidator ID="RangeValidator1" runat="server" ControlToValidate="TextBox1"
ErrorMessage="年龄不能小于 1 或大于 200" MaximumValue="200"MinimumValue="1">
</asp:RangeValidator>
```

4）RegularExpressionValidator 控件

RegularExpressionValidator 控件用于验证相关输入控件的值是否匹配正则表达式指定的模式。

【例 3-17】 使用 RegularExpressionValidator 验证电子邮件的格式是否正确，如图 3-21 所示。

关键步骤如下。

（1）向页面添加一个 TextBox 控件和一个 RegularExpressionValidator 控件。

（2）选中 RegularExpressionValidator 控件，在属性窗体中设置 ControlToValidate 属性为 TextBox1，设置 ErrorToMessage 为“电子邮件地址必须采用 name@domain. xyz 格式”。

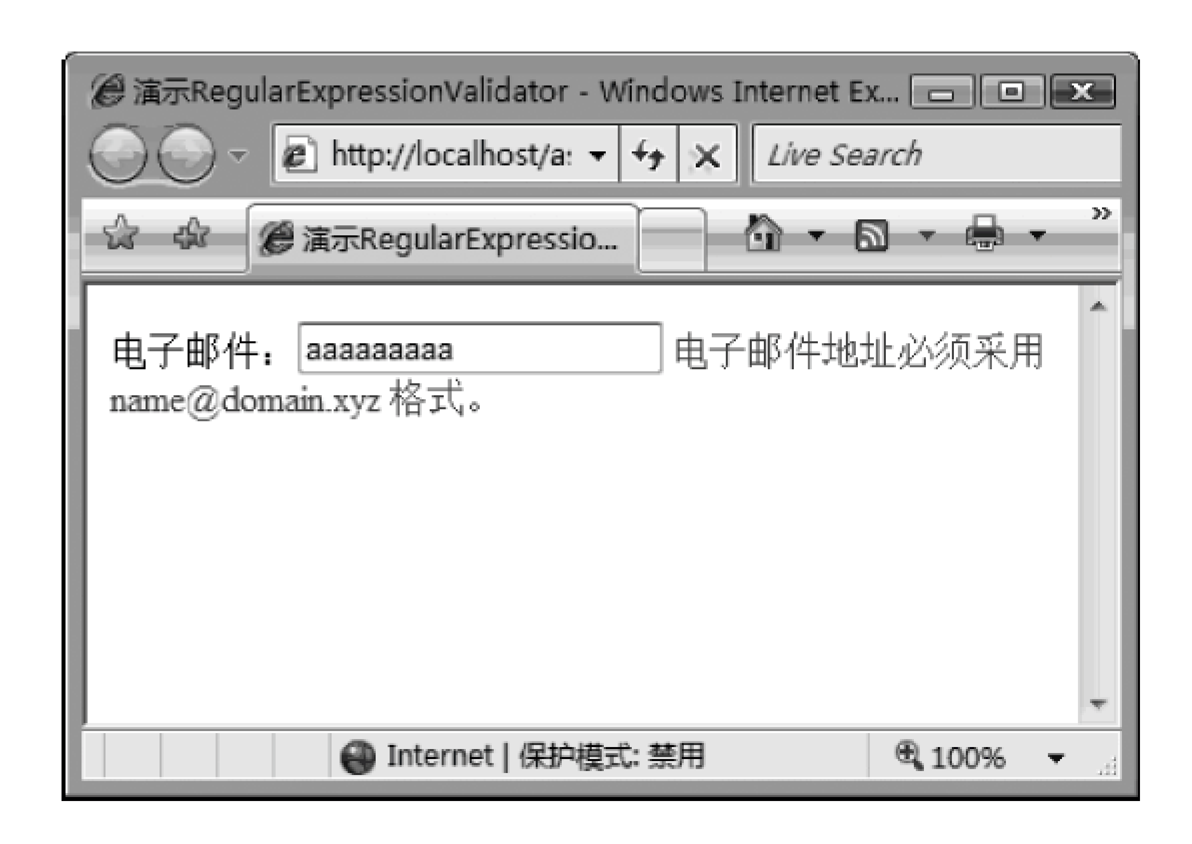


图 3-21　正则表达式验证控件

(3) 在“属性”窗口中单击 ValidationExpression 框中的省略号按钮，打开“正则表达式编辑器”对话框，在“正则表达式编辑器”列表中，单击“Internet 电子邮件地址”。

5) CustomValidator 控件

如果现有的 ASP.NET 验证控件无法满足需求，可以定义一个自定义的服务器端验证函数，然后使用 CustomValidator 控件来调用它。CustomValidator 的语法如下。

```
<asp: CustomValidator id = "控件的名字" runat = "server"
ControlToValidate = "要被验证的控件名字"  OnServerValidate = "服务器端验证函数"
ErrorMessage = "验证错误时的提示信息"  Text = "验证失败时验证控件中显示的文本"
Display = "获取或设置验证控件中错误信息的显示行为">
</asp:CustomValidator >
```

【例 3-18】 使用 CustomValidator 控件来验证用户的出生日期输入的格式是否正确，如图 3-22 所示。

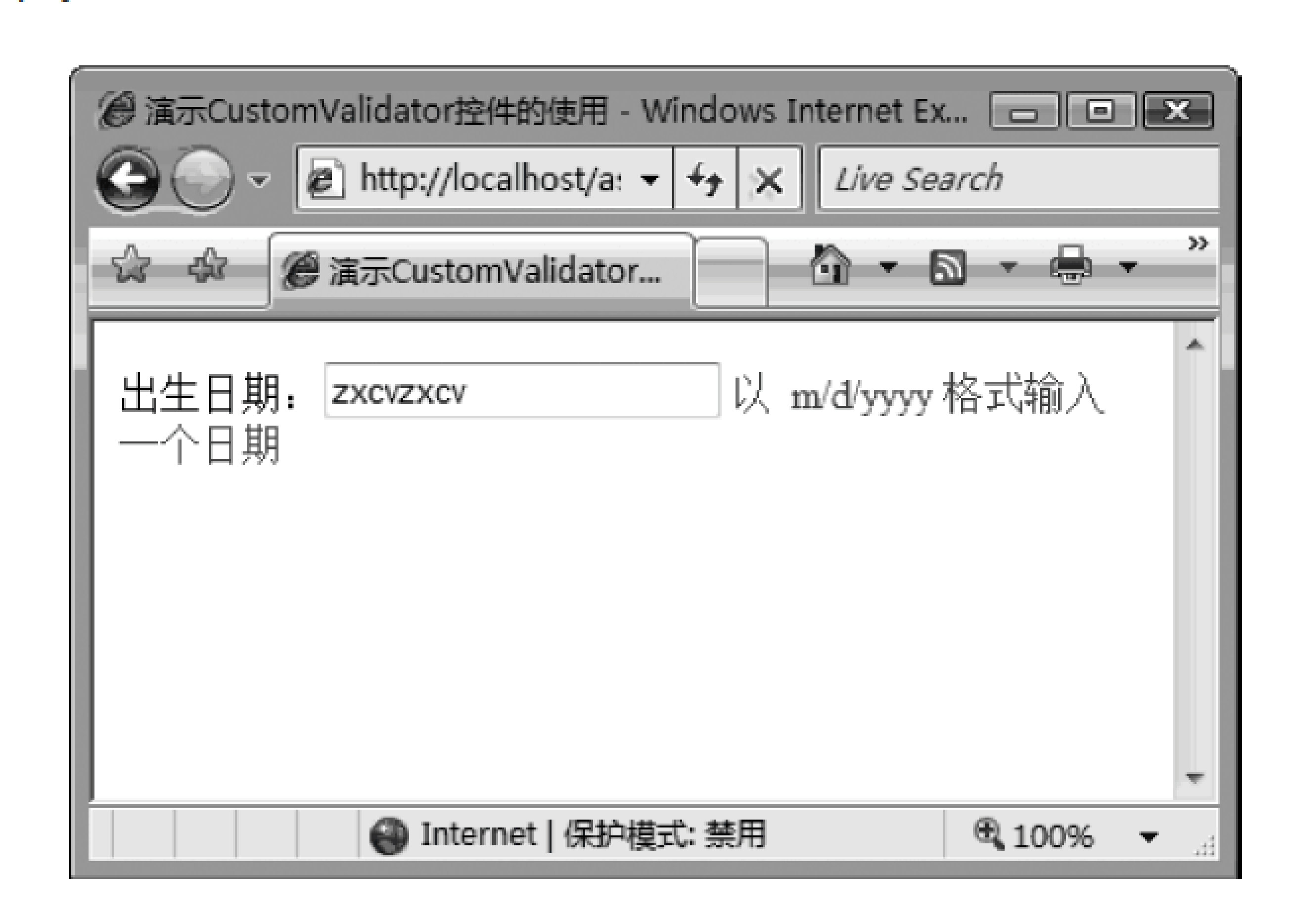


图 3-22　用户自定义验证控件

主要代码如下。

```
<asp:TextBox runat = "server" ID = "TextBox1" TextMode = "SingleLine"></asp:TextBox >
```

```
< asp: CustomValidator  ID  =  " CustomValidator1 "  runat  =  " server "  OnServerValidate  =
"TextValidate"
ControlToValidate = "TextBox1" ErrorMessage = "以 m/d/yyyy 格式输入一个日期">
</asp:CustomValidator >
```

6）ValidationSummary 控件

ValidationSummary 控件用于显示页面中的所有验证错误的摘要。该控件的主要属性有以下几个。

(1) DisplayMode 属性：设置验证摘要的显示模式。

(2) ShowSummary 属性：指定是显示还是隐藏 ValidationSummary 控件。

(3) ShowMessageBox 属性：指定是否显示一个消息对话框显示验证的摘要信息。

(4) HeaderText 属性：获取或设置显示在摘要上方的标题文本。

3.2.4　会员注册页面设计

会员注册页面是网站开发过程中经常用到的，本实例将利用本节所学知识进行会员注册页面设计，本节的会员注册仅完成会员注册信息填写（图 3-23）和信息显示功能（图 3-24）。

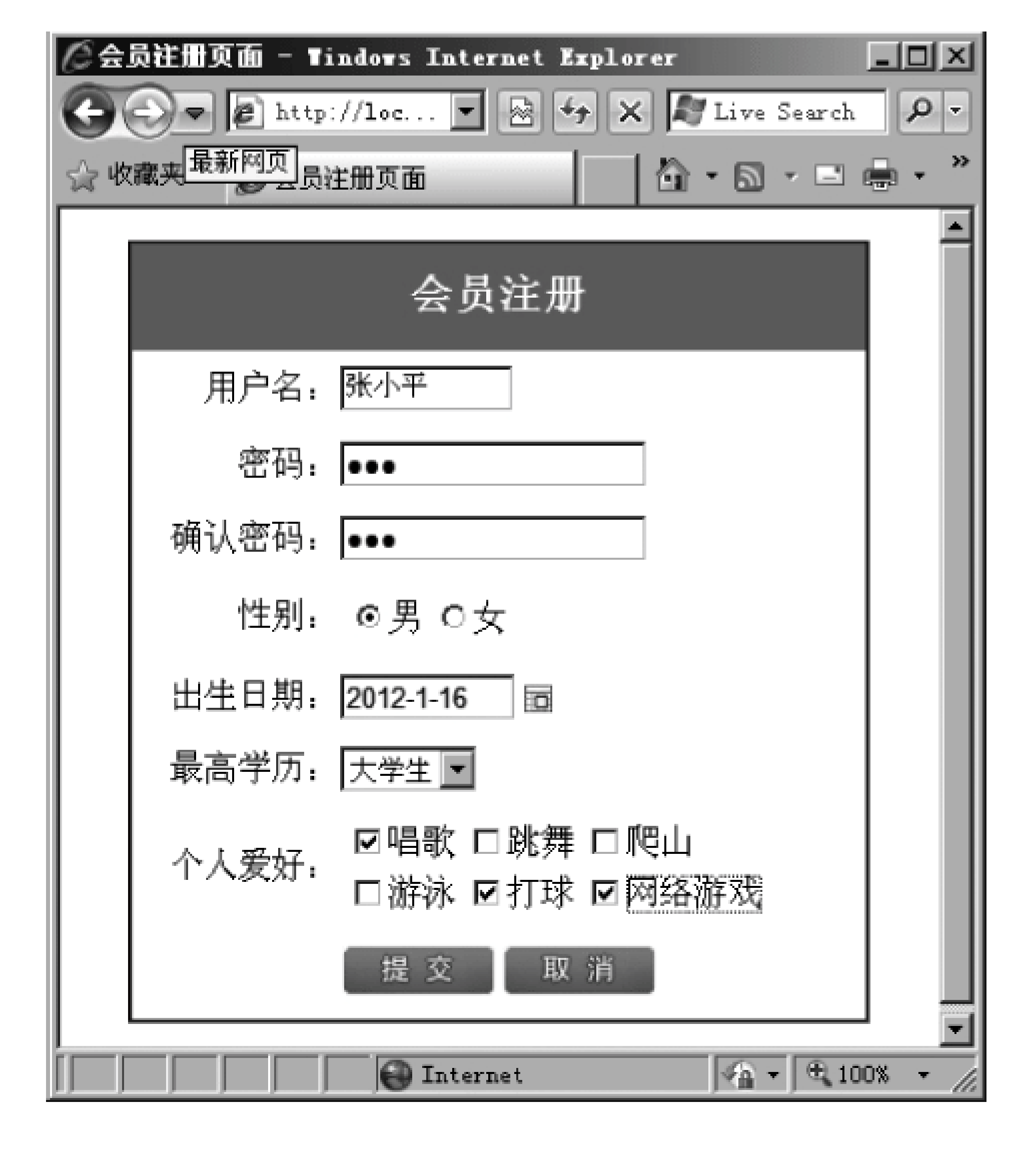


图 3-23　会员注册信息填写

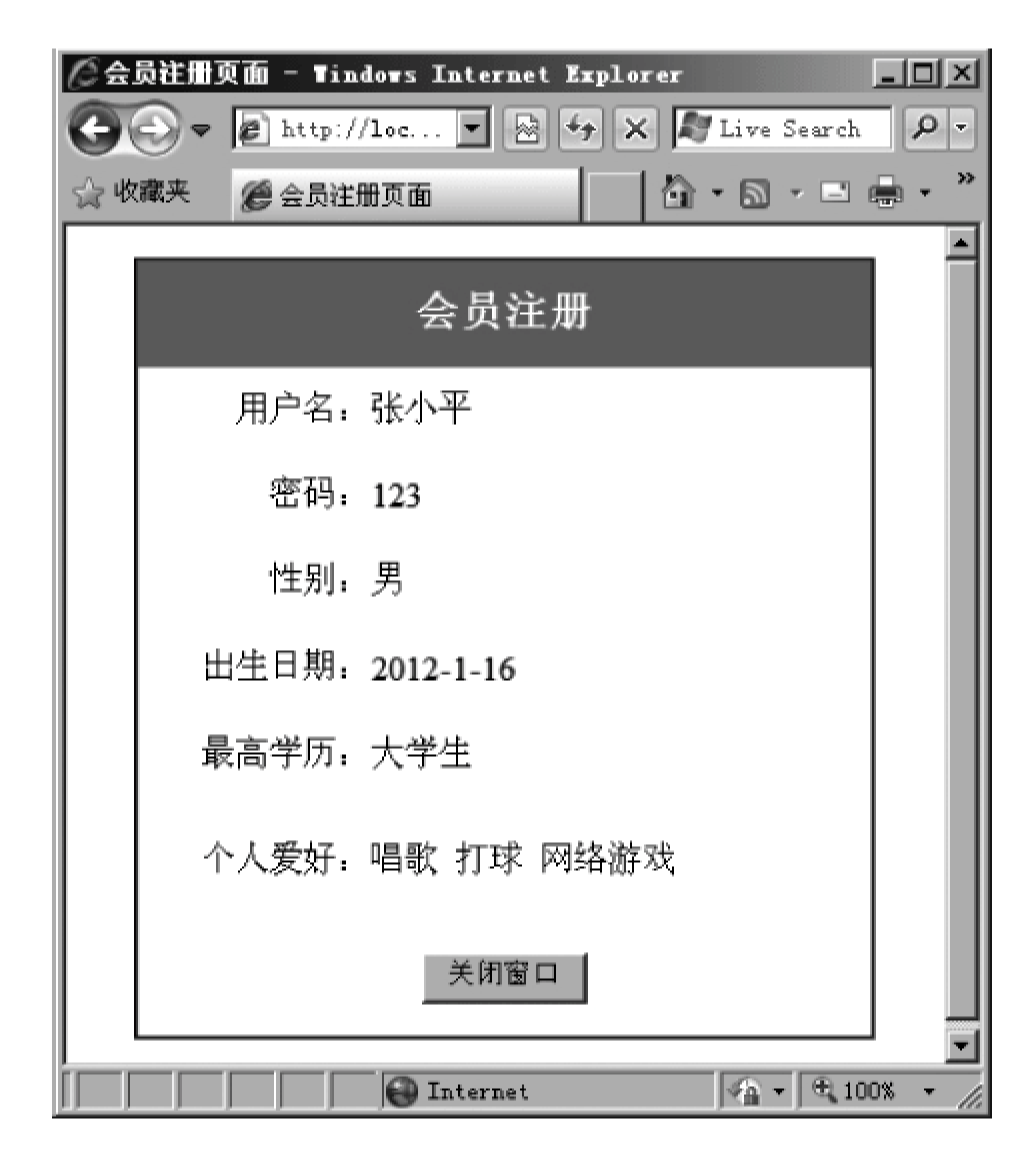


图 3-24　会员注册信息显示

设计步骤如下。

(1) 新建 Addmember.aspx，在页面添加 Panel 控件，设置 ID 为 Panel1。在 Panel1 控件内添加一个 9 行 2 列的 Table 控件。

(2) 将 Table 左边单元格 td 的 align 属性全部设置为 right（即右对齐），右边单元格设置为 left（即左对齐）。将第一行的两个单元格合并，输入“会员注册”，并设置单元格格式。

(3) 左边单元格自上向下依次输入“用户名：”“密码：”“确认密码：”“性别：”“出生

日期：”“最高学历：”和“个人爱好：”。

（4）右边单元格自上向下依次添加相应的服务器控件，并设置其属性，如表 3-8 所示。

表 3-8　会员注册页面的服务器控件

内容	控件 ID	控件类型	说　明
用户名	txtname	TextBox	设置 Width 属性为 82px
密码	txtpwd	TextBox	设置 TextMode 属性为 Password
确认密码	txtpwd2	TextBox	设置 TextMode 属性为 Password
性别	rbtlsex	RadioButtonList	设置 RepeatDirection 属性为 Horizontal，ListItem 为“男”“女”
出生日期	Txtdate	TextBox	设置 Enabled 属性为 False，Width 属性为 83px
	Ibtndate	ImageButton	设置 ImageUrl="～/images/图标.JPG"，onclick="ibtndate_Click"
	clddate	Calendar	设置 onselectionchanged = " clddate _ SelectionChanged "，Visible ="False"
最高学历	ddlschool	DropDownList	设置 ListItem 为研究生、大学生、中学生、小学生
个人爱好	chblove	CheckBoxList	设置 RepeatDirection="Horizontal"，RepeatColumns="3"

（5）将表格最下面一行两个单元格合并，添加两个 ImageButton 控件表示“提交”和“取消”。

（6）按照上面的操作方法，在 Table 后面再添加一个 Panel 控件 Panel2。在 Panel2 内添加 8 行 2 列的 Table 控件，表格第一个单元格和左边一列内容与上面表格内容相同；右边一列单元格里，依次添加 Label 服务器控件，ID 属性依次为 lblname、lblpwd、lblsex、lbldate、lblschool、lbllove 等。

（7）最下面一行两个单元格合并，添加一个 Button 控件，设置 onclick="Button1_Click"和 Text="关闭窗口"。

（8）添加 Page_load 和其他 Button 按钮单击事件代码。

```
protected void Page_Load(object sender, EventArgs e)
{
  Panel2.Visible = false;
  Panel1.Visible = true;
  txtpwd.Attributes["value"] = txtpwd.Text;
  txtpwd2.Attributes["value"] = txtpwd2.Text;
}
protected void ibtndate_Click(object sender, ImageClickEventArgs e)
{clddate.Visible = true;}
protected void clddate_SelectionChanged(object sender, EventArgs e)
{
  txtdate.Text = clddate.SelectedDate.ToShortDateString();
  clddate.Visible = false;
}
protected void btnOk_Click(object sender, ImageClickEventArgs e)
{
```

```
    Panel1.Visible = false;
    Panel2.Visible = true;
    lbldate.Text = txtdate.Text;
    lbllove.Text = "";
    for (int i = 0; i < chblove.Items.Count; i++)
    {
      if (chblove.Items[i].Selected)
      lbllove.Text += " " + chblove.Items[i].Text;
    }
    lblname.Text = txtname.Text;
    lblpwd.Text = txtpwd.Text;
    lblschool.Text = ddlschool.SelectedItem.Text;
    lblsex.Text = rbtlsex.SelectedValue;
}
protected void btnclose_Click(object sender, EventArgs e)
{
    Response.Write("<script>window.close();</script>");
}
protected void btnOk_Click1(object sender, ImageClickEventArgs e)
{
    Panel1.Visible = false;
    Panel2.Visible = true;
    lbldate.Text = txtdate.Text;
    lbllove.Text = "";
    for (int i = 0; i < chblove.Items.Count; i++)
    {
      if (chblove.Items[i].Selected)
      lbllove.Text += " " + chblove.Items[i].Text;
    }
    lblname.Text = txtname.Text;
    lblpwd.Text = txtpwd.Text;
    lblschool.Text = ddlschool.SelectedItem.Text;
    lblsex.Text = rbtlsex.SelectedValue;
}
protected void Ibtncancel_Click(object sender, ImageClickEventArgs e)
{   Response.Redirect("addmember.aspx");}
```

3.3 ASP.NET 内置对象

在使用网站过程中，时常会见到会员管理、网站浏览次数统计、当前网站在线用户人数、在线聊天室和网上投票等内容。在使用网站时，如何进行存储用户信息，并实现跨页面传递呢？这就用到了 ASP.NET 内置对象。

3.3.1 ASP.NET 常用内置对象

ASP.NET 提供了大量的对象类库，在这些类库中包含了许多封装好的内置对象，只需要直接使用这些对象的方法和属性，就能简单、快速地完成很多功能。

内置对象可以在页面上以及页面之间方便地实现获取、输出、传递、保留各种信息等操作，以完成复杂功能。内置对象是对服务器控件很好的补充，进一步扩展了 ASP.NET 程序的功能。

常用的内置对象有 Page、Response、Request、Session、Application、Server 和 Cookie 等。常用内置对象及功能如表 3-9 所示。

表 3-9 常用内置对象及功能

内置对象	功　　能
Page	页面对象，用于整个页面的操作
Request	从客户端获取信息
Response	向客户端输出信息
Session	存储特定用户的信息
Application	存储同一个应用程序中所有用户之间的共享信息
Server	创建 COM 组件和进行有关设置
Cookie	用于保存 Cookie 信息

Page 对象、Request 对象、Response 对象和 Server 对象主要用来进行服务器和客户端浏览器之间的联系；Cookie 对象、Session 对象和 Application 对象则主要用于网站状态管理。

1. Page 对象

Page 对象由 System. Web. UI. Page 类实现，它主要用于处理 ASP.NET 页面的内容。

1）Page 对象工作过程

（1）客户端浏览器向 Web 应用程序进行一个页面的请求。

（2）服务器端 Web 应用程序接收到这个请求，先查看这个页面是否被编译过，如果没有被编译过，就编译这个 Web 页面，然后对这个页面进行实例化产生一个 Page 对象。

（3）Page 对象根据客户请求，把信息返回给 IIS，然后信息由 IIS 返回给客户端浏览器。

在这个过程中，每个页面都被编译成一个类，当有请求时就对这个类进行实例化。

2）Page 对象常用属性、事件和方法

（1）Page 对象常用属性。Page 对象的常用属性如表 3-10 所示。

表 3-10 Page 对象常用属性

属　性	说　　明
IsPostBack	指示该页是否为响应客户端发回而加载，若该属性的值为 True，则表示该页是由于客户端发回而加载的。它是 Page 对象最为重要的属性，该属性常用于判断页面是否是第一次被加载，当页面是第一次加载时，IsPostBack 属性值为 False，否则值为 True
IsValid	指示页验证是否成功。若 IsValid 的值为 True，则意味着网页上的验证控件全部验证成功；否则表示至少有一个控件验证失败
Response	获取与该 Page 对象关联的 HttpResponse 对象

（2）Page 对象常用事件。Page 对象的常用事件如表 3-11 所示。

表 3-11　Page 对象常用事件

事　件	说　　明
PreInit	在页初始化开始时发生
Init	当服务器控件初始化时发生
PreLoad	在页 Load 事件之前发生
Load	当服务器控件加载到 Page 对象中时发生
Unload	当服务器控件从内存中卸载时发生

在 ASP.NET 网页开始载入到被完全写入浏览器的过程中，产生与 Page 对象有关的主要事件有 Init、Load 和 UnLoad 这 3 个。触发顺序如图 3-25 所示。

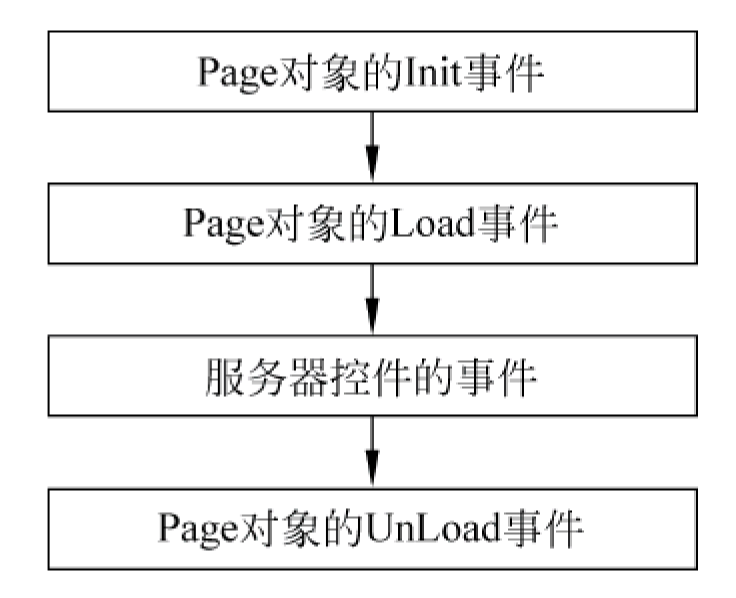


图 3-25　Page 对象的主要事件触发顺序

（3）Page 对象的常用方法。Page 对象的常用方法如表 3-12 所示。

表 3-12　Page 对象的常用方法

方　　法	说　　明
MapPath(virtualPath)	将 virtualPath 指定的虚拟路径转换成实际路径
ResolveUrl(relativeUrl)	将相对地址 relativeUrl 转换为绝对地址
Validate()	执行网页上的所有验证控件
DataBind()	将数据源连接到网页上的服务器控件
Dispose()	强制服务器控件在内存释放之前执行最终的清理工作

【例 3-19】 设计动态添加候选项的页面。当页面初次加载时，“个人爱好”显示“游泳”“唱歌”和“爬山” 3 个选项，下面的文本框里显示“请输入新的选项”。用户在文本框中输入选项内容，并单击“添加”按钮，可以实现选项的添加，效果如图 3-26 所示。

主要代码如下。

```
<div>
个人爱好: <asp:CheckBoxList ID = "ckbtnllove" runat = "server" RepeatDirection =
"Horizontal">
<asp:ListItem>游泳</asp:ListItem>
<asp:ListItem>唱歌</asp:ListItem>
<asp:ListItem>爬山</asp:ListItem>
```



图 3-26　动态添加候选项的页面

```
</asp:CheckBoxList><asp:TextBox ID="txtadd" runat="server"></asp:TextBox>
<asp:Button ID="btnadd" runat="server" Text="添加" /></div>
protected void Page_Load(object sender, EventArgs e)
{   if (!IsPostBack)
    txtadd.Text = "请输入新的选项";
    else
    ckbtnllove.Items.Add(txtadd.Text);
}
```

2. Response 对象

Response 对象由 System.Web.HttpResponse 类实现，主要用于控制对浏览器的输出。它可以用来在页面中输入数据，在页面中跳转，还可以传递各个页面的参数。

Response 对象常用属性和方法如表 3-13 所示。

表 3-13　Response 对象常用属性和方法

名　　称	说　　明
Buffer 属性	设置是否缓冲输出，取值为 True 或 False，默认为 True
ContentType 属性	控制输出的文件类型
Cookies 属性	获取响应 Cookie 集合
Write 方法	Response 对象最常用的方法，用于输出信息到客户端
WriteFile 方法	用于把指定的文件直接写入 HTTP 响应输出流
Redirect 方法	将客户端重定向到新的 URL
Clear 方法	清除缓冲区流中的所有内容输出
End 方法	将当前所有缓冲的输出发送到客户端，停止该页的执行，并引发 EndRequest 事件

【例 3-20】 利用 DropDownList 控件的 SelectedIndexChanged 事件，实现动态改变 LinkButton 控件的显示，并利用 Response 对象的 Redirect 方法实现页面地址重定向，效果如图 3-27 所示。

具体代码如下。

```
<div>
```

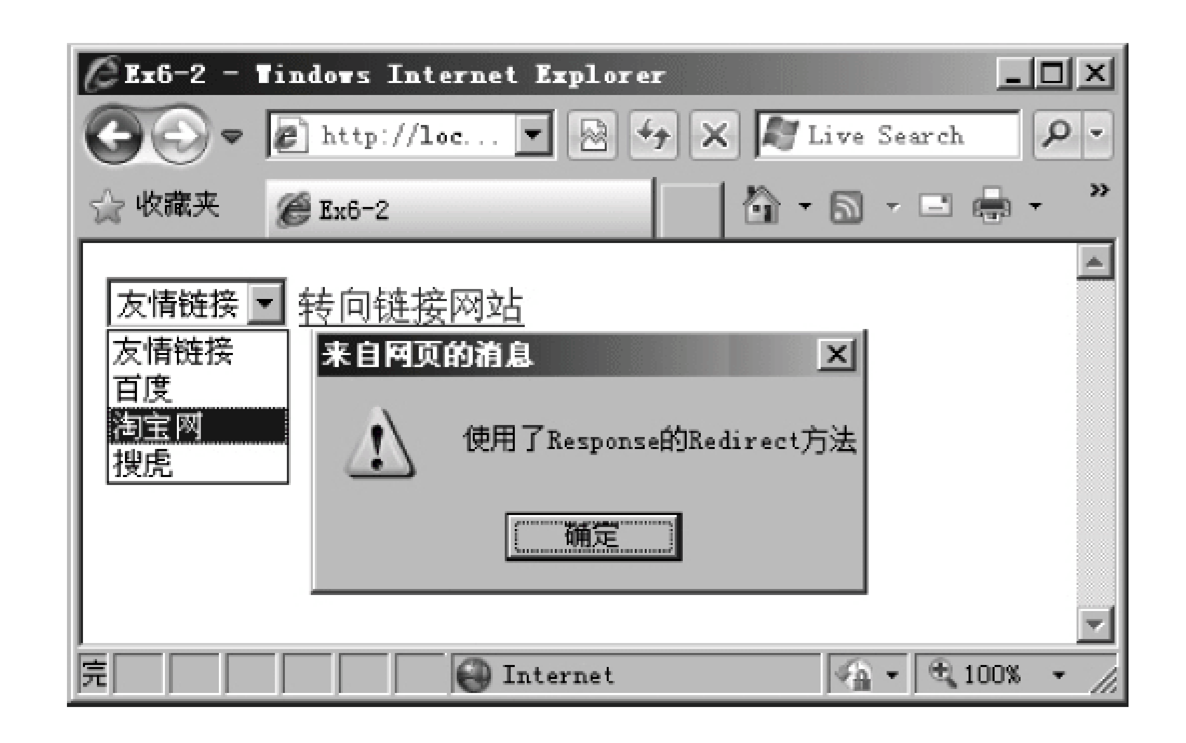


图 3-27 利用 Response 对象的 Redirect 方法实现页面地址重定向

```
<asp: DropDownList ID = " ddlfri " runat = " server " AutoPostBack = " True "
onselectedindexchanged = "ddlfri_SelectedIndexChanged">
<asp:ListItem Value = "Ex6 - 2. aspx">友情链接</asp:ListItem>
<asp:ListItem Value = "http://www. baidu. com">百度</asp:ListItem>
<asp:ListItem Value = "http:// www. taobao. com">淘宝网</asp:ListItem>
<asp:ListItem Value = "http:// www. sohu. com">搜虎</asp:ListItem>
</asp:DropDownList>
<asp:LinkButton ID = "lkbtnfri" runat = "server" onclick = "lkbtnfri_Click">转向链接网站
</asp:LinkButton></div>
protected void ddlfri_SelectedIndexChanged(object sender, EventArgs e)
{   Response. Write("<script>alert('使用了 Response 的 Redirect 方法')</script>");
    lkbtnfri. Text = ddlfri. SelectedItem. Text;}
protected void lkbtnfri_Click(object sender, EventArgs e)
{   Response. Redirect(ddlfri. SelectedValue);}
```

3. Request 对象

Request 对象由 System. Web. HttpRequest 类实现，主要用于获取客户端信息。当用户打开 Web 浏览器并从网站请求 Web 页时，Web 服务器就接收一个 HTTP 请求，此请求包含用户、用户的计算机、页面以及浏览器的相关信息，这些信息将被完整地封装，并通过 Request 对象使用。

Request 对象的常用属性和方法如表 3-14 所示。

表 3-14 Request 对象的常用属性与方法

名称	说明
Form 属性	获取客户端在 Web 表单中所输入的数据集合
QueryString 属性	获取 HTTP 查询字符串变量集合
Cookies 属性	获取客户端发送的 Cookie 集合，用于查看访问者在以前访问本站点时使用的 Cookies
ServerVariables 属性	获取 Web 服务器环境变量的相关信息
Path	返回页面完整的 Web 路径地址，包括页面的文件名称
ApplicationPath	说明被请求的页面位于 Web 应用程序的哪一个文件夹中
PhysicalApplicationPath	返回页面的完整路径，它位于物理磁盘上，不是一个 Web 地址

续表

名　　称	说　　明
Browser 属性	获取或设置有关正在请求的客户端浏览器的功能信息
MapPath 方法	获取当前请求的 URL 虚拟路径映射到服务器上的物理路径
SaveAs 方法	将 HTTP 请求保存到硬盘

【例 3-21】 在网页中显示应用程序的根路径、虚拟目录和浏览器配置的语言设置等基本信息。运行结果如图 3-28 所示。

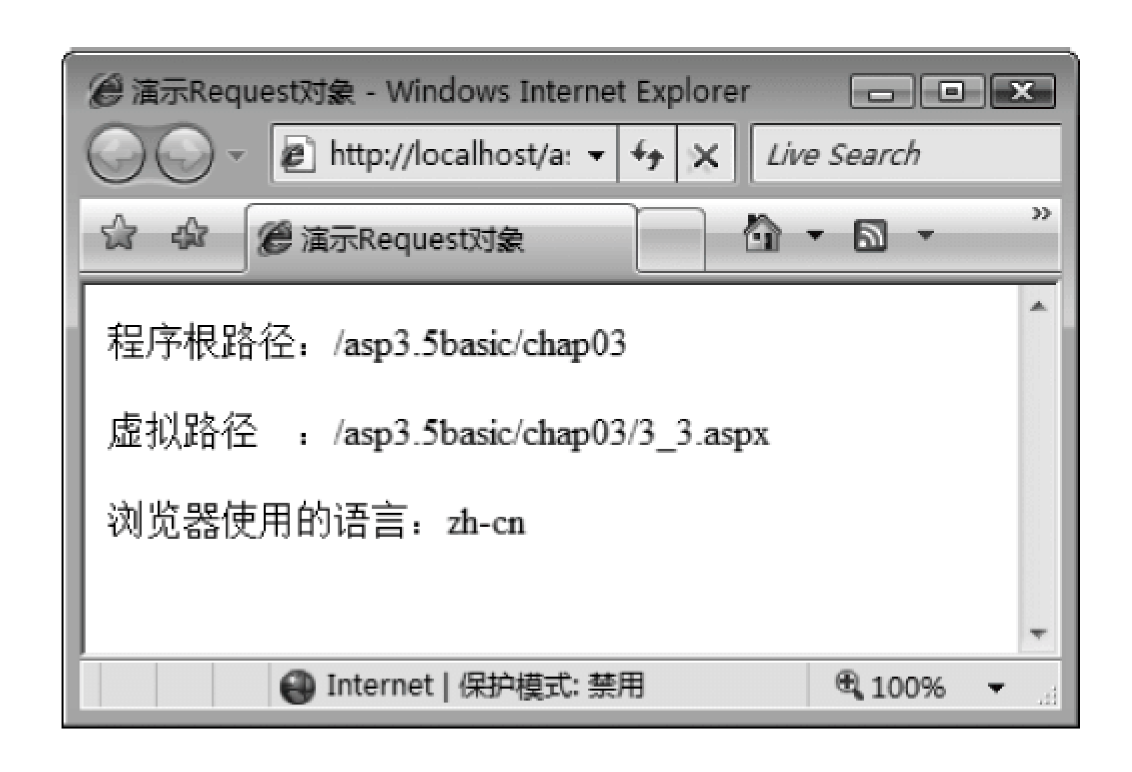


图 3-28　Request 对象的常用属性

主要代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{   if(!IsPostBack)
    {
        Label1.Text = Request. ApplicationPath;
        Label2.Text = Request. Path;
        Label3.Text = Request.UserLanguages[0];
    }
}
```

4. Session 对象

Session 对象由 System. Web. SessionState 类实现，主要用于记载特定用户信息。用户对页面进行访问时，ASP.NET 应用程序会为每一个用户分配一个 Session 对象，即不同用户拥有各自不同的 Session 对象。由于 Session 对象可以在网站的任意一个页面进行访问，所以常用于存储需要跨页面使用的信息。

Session 对象可以为每个用户的会话存储信息。Session 对象中的信息只能被用户自己使用，而不能被网站的其他用户访问，因此可以在不同的页面间共享数据，但是不能在用户间共享数据。

当每个用户首次与服务器建立连接时，服务器就会为其建立了一个 Session，同时服务器会自动为用户分配一个 SessionID，用以标识这个用户的唯一身份。对于每个用户的每次访问 Session 对象是唯一的，此外，Session 对象的有效性是受时间限制的。

1）Session 对象的常用属性和方法

Session 对象的常用属性和方法如表 3-15 所示。

表 3-15 Session 对象的常用属性和方法

名　称	说　明
SessionID 属性	获取会话唯一标识符，存储用户的 SessionID
Timeout 属性	获取并设置在会话状态提供程序终止会话之前各请求之间所允许的时间（以 min 为单位），默认为 20min
Abandon 方法	取消当前会话，清除 Session 对象

Session 对象具有两个事件：Session_OnStart 事件和 Session_OnEnd 事件。Session_OnStart 事件在创建一个 Session 时触发，Session_OnEnd 事件在用户 Session 结束时被调用。

2）使用 Session 对象

（1）使用 Session 对象保存信息。

【基本语法】

```
Session["键名"] = 值;
```

或

```
Session.Add("键名",值);
```

（2）按名称获取会话状态中的值。

【基本语法】

```
变量 = Session["键名"];
```

（3）删除会话状态集合中的项。

【基本语法】

```
Session.Remove("键名")
```

（4）清除会话状态中的所有值。

【基本语法】

```
Session.RemoveAll()
```

或

```
Session.Clear()
```

（5）取消当前会话。

【基本语法】

```
Session.Abandon()
```

（6）设置会话状态的超时期限，以分钟为单位。

【基本语法】

```
Session.TimeOut = 数值
```

【例 3-22】 利用 Session 对象实现网站后台登录的身份验证。在第一个页面中，用户输入用户名和密码，单击"后台管理"按钮后，将用户名和密码信息保存至 Session 对象中。在第二个页面中先利用 Session["user"]来判断用户是否已登录，若登录则出现"用户注销"按钮；否则出现无权访问的提示。同时，单击"用户注销"按钮实现 Session 对象信息清除。效果如图 3-29 所示。

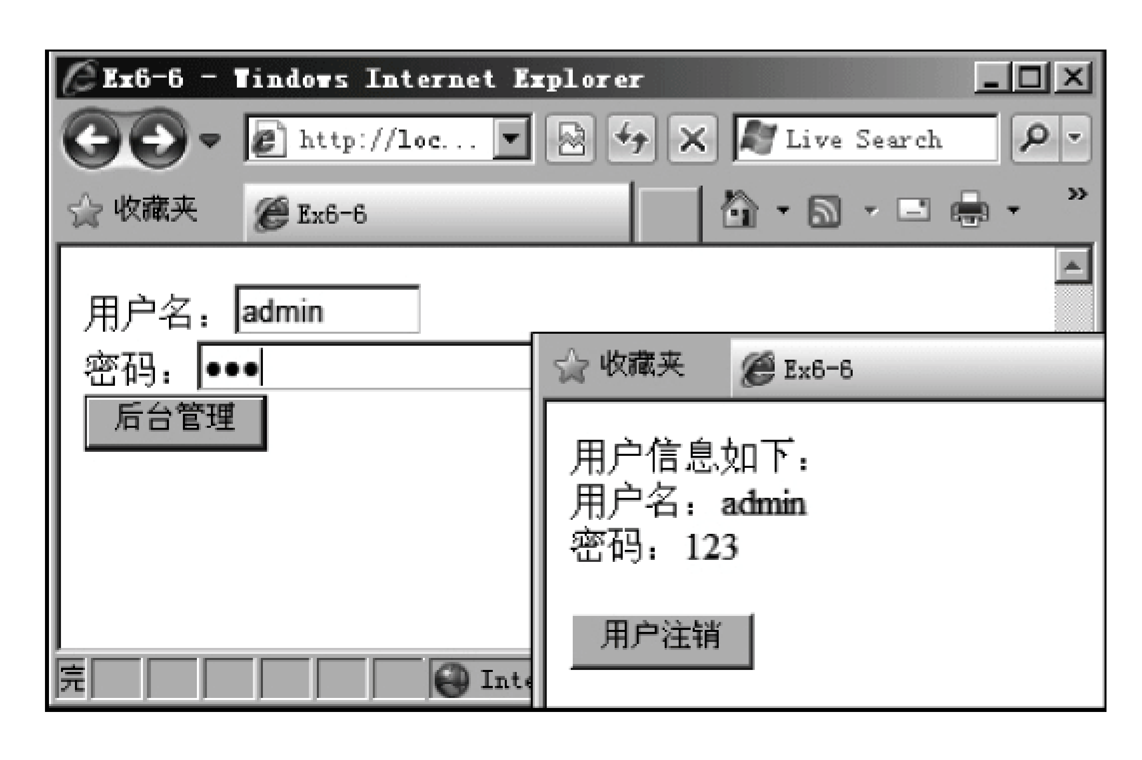


图 3-29 Session 对象

第一个页面主要代码如下。

```
<div>用户名：<asp:TextBox ID = "txtname" runat = "server" Width = "80px"></asp:TextBox>
密码：<asp:TextBox ID = "txtpwd" runat = "server" TextMode = "Password"></asp:TextBox>
<asp:Button ID = "Button1" runat = "server" Text = "后台管理" onclick = "Button1_Click"/></div>
protected void Button1_Click(object sender, EventArgs e)
{
    Session["user"] = txtname.Text;
    Session["pwd"] = txtpwd.Text;
    Response.Redirect("Ex6 - 6(2).aspx");
}
```

第二个页面主要代码如下。

```
<div><asp:Label ID = "lblmes" runat = "server" Text = "Label"></asp:Label>
<asp:HyperLinkID = "hplback" runat = "server" NavigateUrl = "~/Ex6 - 6.aspx" Visible =
"False">返回上一页</asp:HyperLink>
<asp:ButtonID = "btnquit" runat = "server" onclick = "btnquit_Click" Text = "用户注销"
Visible = "False" />
</div>
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["user"] != null && Session["user"].ToString() != "")
    {
```

```
            lblmes.Text = "用户信息如下:<br>用户名:" + Session["user"].ToString() + "<br>
            密码:" + Session["pwd"].ToString();
            btnquit.Visible = true;
            Session.Timeout = 10;
        }else
        {
            lblmes.Text = "你无权进入后台管理!6 秒后自动返回上页。<br>或单击下面的链接。";
            hplback.Visible = true;
            Response.Write("<script>setTimeout('window.history.back()', 6000)</script>");
        }
    }
    protected void btnquit_Click(object sender, EventArgs e)
    {   Session.Abandon();
        Response.Redirect("Ex6 - 6(2).aspx");
    }
```

5. Application 对象

Application 对象由 System. Web. HttpApplication 类实现，主要用于存储网站的共享信息。与 Session 对象存储信息的方式类似，Application 对象也是将用户信息存储在服务器中。Application 中的信息可以被网站的所有页面访问，因此可以在不同的用户间共享数据。

在 ASP.NET 中，使用 Application 对象代表 ASP.NET Web 应用程序的运行实例。一个 Web 站点可以包含不止一个 ASP.NET 应用程序，而每个 ASP.NET 应用程序的运行实例都可以由一个 Application 对象来表达。可以将任何对象作为全局变量存储在 Application 对象中。

ASP.NET 应用程序是单个 Web 服务器上的某个虚拟目录及其子目录范围内的所有文件、页、处理程序、模块和代码的总和。

1）Application 对象与 Session 对象的不同

(1) Application 对象是一个公用变量，允许应用程序的所有用户使用；而 Session 对象只允许某个特定的用户使用。

(2) Application 对象的生命周期止于网站 IIS 关闭或者 Clear()方法清除；而 Session 对象的生命周期止于用户页面的关闭或者 Abandon()方法清除。

2）Application 对象的常用事件

Application 的常用事件如表 3-16 所示。

表 3-16 Application 对象的常用事件

事件名称	说明
Application_Start	在应用程序启动时激发
Application_BeginRequest	在每个请求开始时激发
Application_ AuthenticateRequest	尝试对使用者进行身份验证时激发
Application_Error	在发生错误时激发
Application_End	在应用程序结束时激发

Application 对象的事件在 Global.asax 文件中进行处理，添加用户自定义代码。

3）Application 对象的常用方法

由于多个用户可以共享一个 Application 对象，为了保证用户在修改 Application 对象值时的资源同步访问，需要使用 Application 对象的 Lock 和 Unlock 方法进行对象的加锁和解锁。解决对 Application 对象的访问同步问题，一次只允许一个线程访问应用程序状态变量。

锁定：Application.Lock()。

解锁：Application.Unlock()。

注意：*Lock 方法和 UnLock 方法应该成对使用。*

4）Application 对象的基本语法

（1）使用 Application 对象保存信息。

**【基本语法】**

```
Application["键名"] = 值;
```

或

```
Application.Add("键名",值);
```

（2）获取 Application 对象信息。

**【基本语法】**

```
变量名 = Application["键名"];
```

或

```
变量名 = Application.Get("键名");
```

（3）更新 Application 对象的值。

**【基本语法】**

```
Application.Set("键名",值);
```

或

```
Application["键名"] = 值
```

（4）删除一个键。

**【基本语法】**

```
Application.Remove("键名",值)
```

（5）删除所有键。

**【基本语法】**

```
Application.RemoveAll()
```

或

```
Application.Clear()
```

【例 3-23】 使用 Application 对象实现网站访问数量统计,效果如图 3-30 所示。

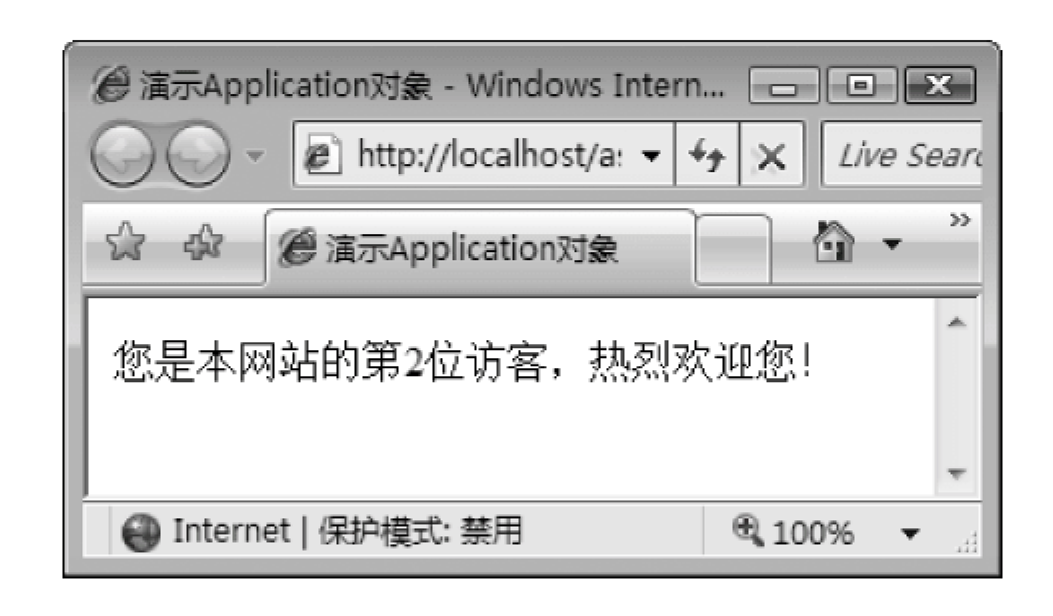


图 3-30 使用 Application 对象实现网站访问数量统计

主要代码如下。

```
<div>您是本网站的第<asp:Label ID = "lblnum" runat = "server" ForeColor = "Red"></asp:
Label>
位访客,热烈欢迎您!</div>
protected void Page_Load(object sender, EventArgs e)
{
    if (Application["usernum"] == null)
    Application["usernum"] = 1;
    else{
        Application.Lock();
        Application["usernum"] = (Int32)Application["usernum"] + 1;
        Application.UnLock();
    }
    lblnum.Text = Application["usernum"].ToString();
}
```

6. Cookie 对象

Cookie 对象由 System.Web.HttpCookie 类实现,主要用于客户端存储用户个人信息。Cookie 对象与 Session、Application 对象类似,是一种集合对象,都是用于保存数据。

Cookie 就是 Web 服务器保存在用户硬盘上的一段文本。Cookie 允许一个 Web 站点在用户的计算机上保存信息并且随后再取回它。信息的片段以“键/值”对的形式存储。

Cookie 是保存在客户机硬盘上的一个文本文件,可以存储有关特定客户端、会话或应用程序的信息。

有两种类型的 Cookie:会话 Cookie(Session Cookie)和持久性 Cookie。前者是临时性的,一旦会话状态结束它将不复存在;后者则具有确定的过期日期,在过期之前 Cookie 在用户的计算机上以文本文件的形式存储。

Cookie 对象的常用属性和方法如表 3-17 所示。

表 3-17 Cookie 对象的常用属性和方法

名　称	说　明
Name 属性	获取 Cookie 变量的名称
Value 属性	获取或设置 Cookie 对象的值
Count 属性	获取 Cookies 集合中 Cookie 对象的数量
Expires 属性	设置 Cookie 对象的生命周期,默认为 1000min;当值小于等于 0 时,生命周期结束
Add 方法	创建新对象并将其添加到 Cookies 集合中

Cookie 对象不隶属于 Page 对象,分别属于 Request 和 Response 对象,每一个 Cookie 变量都由 Cookies 对象所管理。

(1) 要保存一个 Cookie 变量,需要通过 Response 对象的 Cookies 集合。


【基本语法】

```
Response.Cookies["变量名"].Value = 值;
```

(2) 读取 Cookie 对象时,需要使用 Request 对象。


【基本语法】

```
变量 = Request.Cookies["变量名"].Value;
```

(3) 指定 Cookie 过期日期。


【基本语法】

```
Response.Cookies["UserName"].expires = new DateTime(2015,8,30);
```

或

```
Response.Cookies["UserName"].Expires = DateTime.Now.AddDays(30);
```

【例 3-24】 使用 Cookie 对象实现用户登录信息自动填充。当用户第二次使用该网站时,用户名信息会自动输入,从而方便用户。用户单击“清除 Cookie”按钮时,实现 Cookie 对象中的用户信息清除,效果如图 3-31 所示。

主要代码如下。

```
<div>用户名: <asp:TextBox ID = "txtname" runat = "server" Width = "88px"></asp:TextBox>
密码: <asp:TextBox ID = "txtpwd" runat = "server" TextMode = "Password"></asp:TextBox>
<asp:Button ID = "btnsave" runat = "server" Text = "写入 Cookies" onclick = "btnsave_Click" />
<asp:Button ID = "btnclear" runat = "server" onclick = "btnclear_Click" Text = "清除 Cookie" />
</div>
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.Cookies["mycookie"]!= null)
    txtname.Text = Request.Cookies["mycookie"].Value;
```

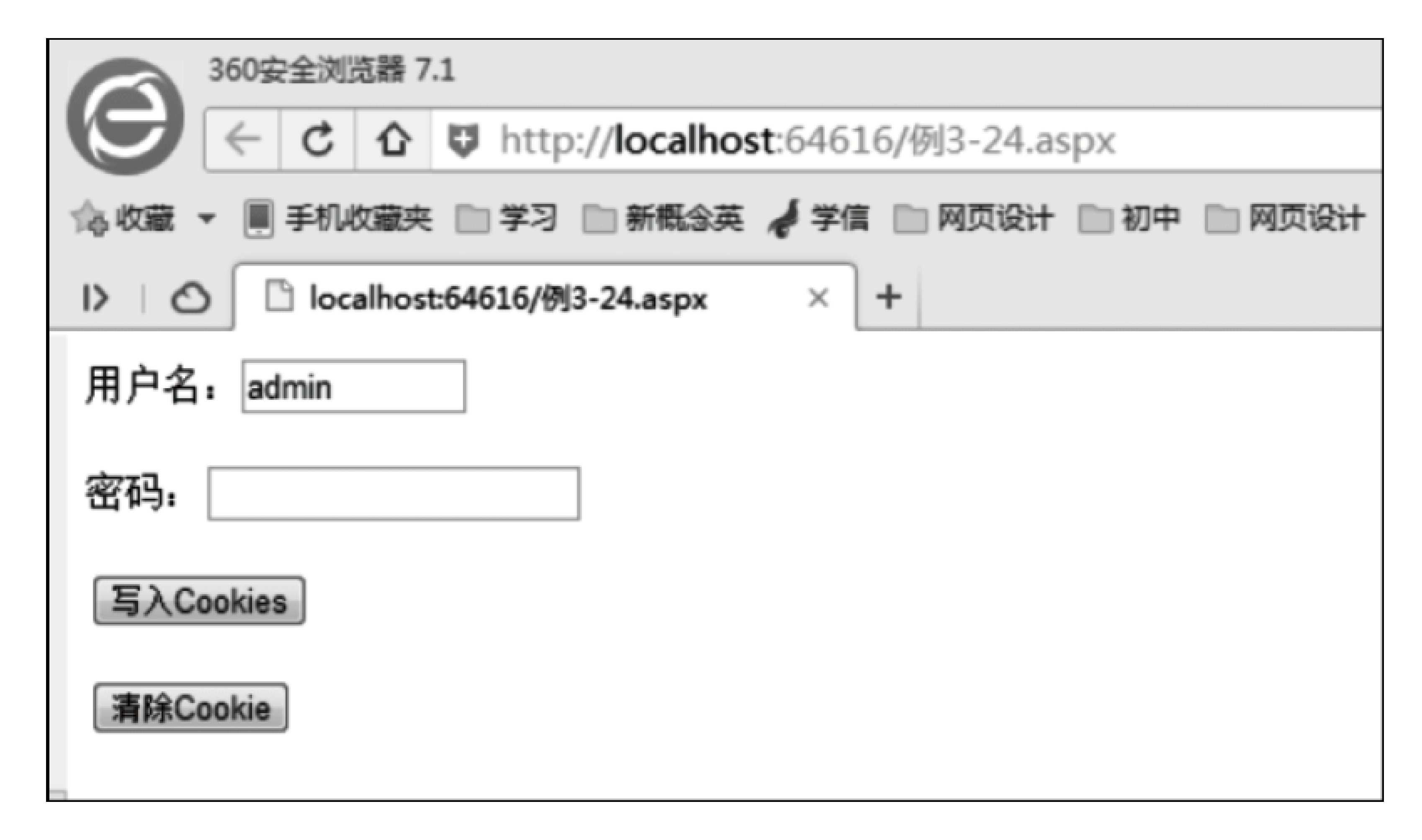


图 3-31　使用 Cookie 对象实现用户登录信息自动填充

```
}
protected void btnsave_Click(object sender, EventArgs e)
{
    Response.Cookies["mycookie"].Value = txtname.Text;
    Response.Cookies["mycookie"].Expires = DateTime.Now.AddDays(30);
}
protected void btnclear_Click(object sender, EventArgs e)
{
    HttpCookie acookie;
    string ckname;
    int cknum = Request.Cookies.Count;
    for (int i = 0; i < cknum; i++)
    {   ckname = Request.Cookies[i].Name;
        acookie = new HttpCookie(ckname);
        acookie.Expires = DateTime.Now.AddDays(-1);
        Response.Cookies.Add(acookie);
    }
    Response.AddHeader("Refresh", "0");
}
```

7. Server 对象

Server 对象提供对服务器上的方法和属性进行访问，其类名称是 HttpServerUtility。Server 对象的主要属性有以下两个。

(1) MachineName：获取服务器的计算机名称。

(2) ScriptTimeout：获取和设置请求超时(以秒计)。

Server 对象的常用方法如表 3-18 所示。

表 3-18 Server 对象的常用方法

方　法	说　明
CreateObject	创建 COM 对象的一个服务器实例
Execute	执行当前服务器上的另一个 aspx 页，执行完该页后再返回本页继续执行
HtmlEncode	对要在浏览器中显示的字符串进行 HTML 编码并返回已编码的字符串
HtmlDecode	对 HTML 编码的字符串进行解码，并返回已解码的字符串
MapPath	返回与 Web 服务器上指定虚拟路径相对应的物理文件路径
Transfer	终止当前页的执行，并为当前请求开始执行新页
UrlEncode	将代表 URL 的字符串进行编码，以便通过 URL 从 Web 服务器到客户端进行可靠的 HTTP 传输
UrlDecode	对已被编码的 URL 字符串进行解码，并返回已解码的字符串
UrlPathEncode	对 URL 字符串的路径部分进行 URL 编码，并返回已编码的字符串

3.3.2 综合实例：在线聊天室

相信许多人对聊天室并不陌生，它是网站实现用户互动的主要手段之一。本综合实例通过运用 Session、Application 和 Cookie 等 ASP.NET 内置对象，实现在线聊天室的开发。

1. 数据库表设计

(1) 启动 Access 数据库，新建数据库，命名为 mychat.mdb。

(2) 通过“新建表”命令，创建用户信息表 chatmem。表中字段有用户编号 mid(自动编号)、昵称 mname(文本，10 个字符长度)、密码 mpwd(文本，8 个字符长度)。其中，mid 为主关键字。

(3) 输入部分用户信息，如“happyday、222”“redink、111”等。

(4) 检查网站的“解决方案资源管理器”窗口，是否存在 App_Data 系统文件夹。如果不存在，用户可以通过右击项目，选择快捷菜单中的“添加 ASP.NET 文件夹(s)”→App_Data 命令创建。

(5) 将建好的数据库文件 mychat.mdb 移动到 App_Data 系统文件夹中。

(6) 检查网站的“解决方案资源管理器”窗口，是否存在 Web 配置文件 Web.config。如果不存在，用户可以通过右击项目，选择快捷菜单中的“添加新项”命令。在“添加新项”对话框中选择“Web 配置文件”模板，并将文件命名为 Web.config，单击“添加”按钮。

(7) 在“解决方案资源管理器”窗口中，双击打开 Web.config，找到<appSettings/>节。把<appSettings/>修改为以下代码。

```
<appSettings>
<addkey = "strcon"
value = "Provider = Microsoft.Jet.OLEDB.4.0;Data Source = |DataDirectory|mychat.mdb"/>
</appSettings>
```

(8) 在“解决方案资源管理器”窗口中，右击项目，选择快捷菜单中的“添加新项”命令。在“添加新项”对话框中选择“全局应用程序类”模板，并将文件命名为 Global.asax，

单击“添加”按钮。

(9) 在“解决方案资源管理器”窗口中，双击打开 Global. asax，在 Application_Start 事件中，输入以下代码。

```
void Application_Start(object sender, EventArgs e)
{    Application["mcount"] = 0;
     Application["chatcon"] = "";
     Application["userlist"] = "所有人";
     Application.UnLock();
}
```

(10) 在 Session_Start 事件中，输入以下代码。

```
void Session_Start(object sender, EventArgs e)
{    Application.Lock();
     Application["mcount"] = Convert.ToInt32(Application["mcount"].ToString()) + 1;
     Application.UnLock();
}
```

(11) 在 Session_End 事件中，输入以下代码。

```
void Session_End(object sender, EventArgs e)
{    Application.Lock();
     Application["mcount"] = Convert.ToInt32(Application["mcount"].ToString()) - 1;
     Application.UnLock();
}
```

(12) 在“解决方案资源管理器”窗口中，右击项目，选择快捷菜单中的“添加新项”命令添加一个 Web 窗体，命名为 chatlogin. aspx。

(13) 在页面中添加一个 6 行 2 列的表格，将表格中的第 1、2、5 和 6 行单元格进行合并。在第 1 行单元格中输入“聊天室登录”，并设置单元格格式。

(14) 在第 2 行单元格中输入“欢迎访问聊天室，当前在线人数：”，并在文本后添加一个 Label 控件，设置 ID 属性为 lblnum。

(15) 在第 3 行左侧单元格中输入“昵称：”；右侧单元格中添加 TextBox 控件，设置 ID 属性为 txtname；在右侧添加 RequiredFieldValidator 验证控件，设置 ID 属性为 rqcname，ControlToValidate 属性为 txtname，ErrorMessage 属性为“用户名必须输入”。

(16) 在第 4 行左侧单元格中输入“密码：”；右侧单元格中添加 TextBox 控件，设置 ID 属性为 txtpwd，TextMode 属性为 Password；在右侧添加 RequiredFieldValidator 验证控件，设置 ID 属性为 rqcpwd，ControlToValidate 属性为 txtpwd，ErrorMessage 属性为“密码必须输入”。

(17) 在第 5 行单元格中添加一个 RequiredFieldValidator 复选框控件，设置 ID 属性为 ckbrem，Text 属性为“记录我的信息”。

(18) 在第 6 行单元格中添加两个 Button 控件。第 1 个 Button 控件的 ID 属性为 btnlogin，Text 属性为“登录”；第 2 个 Button 控件的 ID 属性为 btncancel，Text 属性为“取消”。

(19) 双击“登录”控件,输入 btnlogin_Click 单击事件,代码如下。

```
protected void btnlogin_Click(object sender, EventArgs e)
{   string uname = txtname.Text.Trim();
    string upwd = txtpwd.Text.Trim();
    string strcon = System.Configuration.ConfigurationManager.AppSettings["strcon"].
    ToString();
    OleDbConnection conn = new OleDbConnection(strcon);
    string sql0 = "select count( * ) from chatmem where mname = '" + uname.ToLower() + "' and
    mpwd = '" + upwd.ToLower() + "'";
    conn.Open();
    OleDbCommand ocmd = new OleDbCommand(sql0, conn);
    if (Convert.ToInt32(ocmd.ExecuteScalar()) > 0)
    {  if (ckbrem.Checked)
       { Response.Cookies["ckname"].Value = uname;
         Response.Cookies["ckname"].Expires = DateTime.Now.AddDays(15);
       }
       Session["uname"] = uname;
       Application["userlist"] += "," + uname;
       Response.Redirect("chatmain.aspx");
    } else
    Response.Write("< script > alert('用户信息不正确!');</script>");
}
```

(20) 双击“取消”按钮,输入 btncancel_Click 单击事件,代码如下。

```
protected void btncancel_Click(object sender, EventArgs e)
{    Response.AddHeader("Refresh", "0");}
```

(21) 双击页面空白处,输入以下代码。

```
protected void Page_Load(object sender, EventArgs e)
{   lblnum.Text = Application["mcount"].ToString();
    if (!IsPostBack)
    { if (Request.Cookies["ckname"] != null)
      txtname.Text = Request.Cookies["ckname"].Value;}}
}
```

(22) 完成上述操作后,保存文件,按 F5 键运行,效果如图 3-32 所示。

(23) 在“解决方案资源管理器”窗口中,右击项目,选择快捷菜单中的“添加新项”命令,添加一个 Web 窗体,命名为 chatmain.aspx。

(24) 在页面中添加一个 4 行 1 列的表格。在第 1 个单元格中输入“在线聊天室”,并设置单元格格式。

(25) 在第 2 个单元格中添加一个 Label 控件,设置其 ID 属性为 lblchat,宽度和表格宽度一致,以及背景色等格式,用于显示聊天内容。

(26) 在第 3 个单元格中添加一个 Label 控件,设置其 ID 属性为 lblname,用于显示当前用户昵称信息,对应 Session["uname"]对象。再添加一个 DropDownList 控件,设置其 ID 属性为 ddluser,用于显示当前在线用户昵称信息,对应 Application["userlist"]



图 3-32 聊天室登录

对象。

(27) 在第 4 个单元格中添加一个 TextBox 控件，设置其 ID 属性为 txtme，TextMode 属性为 MultiLine，以及控件宽度和高度等属性，用于输入聊天内容。

(28) 在 txtme 控件右侧添加一个 Button 控件，设置其 ID 属性为 btnsend，Text 属性为“发送”。

(29) 双击页面空白处，打开后台代码文件 chatmain.aspx.cs，输入以下 Page_Load 事件代码。

```
protected void Page_Load(object sender, EventArgs e)
{                                              //判断页面是否为第 1 次加载
    if (!IsPostBack)
    {//判断用户是否登录
        if (Session["uname"] != null)
        {   lblname.Text = Session["uname"].ToString() + " 对";
            //用于显示当前在线用户昵称信息
            string[] userlist = Application["userlist"].ToString().Split(',');
            for (int i = 0; i < userlist.Length; i++)
            ddluser.Items.Add(userlist[i]);
            //显示聊天内容
            lblchat.Text = Application["chatcon"].ToString();
        }
        else
        Response.Redirect("chatlogin.aspx");
    }
}
```

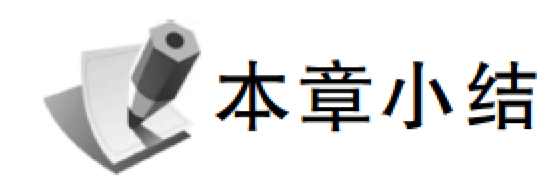
(30) 双击页面中的“发送”按钮 btnsend,添加以下 btnsend_Click 事件代码。

```
protected void btnsend_Click(object sender, EventArgs e)
{   string user = Session["uname"].ToString();
    string touser = ddluser.SelectedValue;
    //采用加锁机制,更新 Application 对象内容
    Application.Lock();
    Application["chatcon"] += "<br>" + user + "对" + touser + "说:" + txtme.Text;
    Application.UnLock();
    lblchat.Text = Application["chatcon"].ToString();
}
```

(31) 保存文件,完成在线聊天室,最终程序运行效果如图 3-33 所示。



图 3-33　显示用户发表的内容



本章小结

本章主要介绍了 Web 窗体、ASP.NET 服务器控件和 ASP.NET 内置对象。重点掌握 ASP.NET 服务器控件和 ASP.NET 内置对象的常用属性、方法和事件。



思考与练习

1. 选择题

(1) 用于在页面上显示文本的控件是(　　)。

A. Label　　B. TextBox　　C. Button　　D. LinkButton

(2) 下面(　　)是单选按钮。

A. ImageButton　　B. LinkButton　　C. RadioButton　　D. BulletedList

(3) CheckBox 是常用的控件,它是指(　　)。

A. 列表框　　B. 文本框　　C. 复选框　　D. 标签

(4) 用于在 ASP.NET 页面上显示图像的控件是(　　)。

A. BorderColor　　B. bgColor　　C. RadioButton　　D. Image

(5) AccessKey 的功能是(　　)。

A. 变量　　B. 存取键　　C. 关键字　　D. 快捷键

(6) 可使用户能够方便地在网站的不同页面之间实现跳转的控件是(　　)。

A. CausesValidation　　B. HyperLink

C. Checked　　D. SelectedIndex

(7) 判断用户通过输入界面中的文本框控件输入的信息是否是偶数,要求用验证控件实现,需要选择(　　)验证控件。

A. RequiredFieldValidator　　B. RangeValidator

C. CompareValidator　　D. CustomerValidator

(8) 当整个页面被浏览器读入时触发的事件是(　　)。

A. Page_Load　　B. Page_Unload　　C. Page_Init　　D. Click

(9) 下面(　　)文件是以 ascx 为扩展名的。

A. 普通的 Web 页面　B. 母板页　　C. 用户控件　　D. 外观文件

(10) 下列(　　)验证控件用于强制用户必须输入某一范围内的信息。

A. RequiredFieldValidator　　B. RangeValidator

C. RegularExpressionValidator　　D. CustomValidator

(11) 通过(　　)属性可以设置 Session 对象的失效时间。

A. Timeout　　B. Timeoff　　C. Timeend　　D. Timeover

(12) 能实现网页转向的方法是(　　)。

A. Response. Appendheader()　　B. Response. Clearheaders()

C. Response. Redirect()　　D. Response. Addheader()

(13) Session 对象的默认有效期为(　　)min。

A. 10　　B. 15

C. 20　　D. 应用程序从启动到结束

(14) 如果需要确保用户输入大于 30 的值,应该使用(　　)验证控件。

A. RequiredFieldValidator　　B. CompareValidator

C. RangeValidator　　D. RegularExpressionValidator

(15) RegularExpressionValidator 控件的功能是(　　)。

A. 用于验证规则

B. 用于展示验证结果

C. 用于判断输入的内容是否满足指定的范围

D. 用于判断输入的内容是否符合指定的格式

2. 简答题

(1) 在网站的页面中添加控件有哪两种方法?

(2) 进入代码编辑窗口有哪两种方法?

第4章

数据库技术

数据库技术是使用计算机进行数据管理的核心技术，使用 ASP.NET 进行应用程序开发时，必须掌握和应用数据库原理及技术。本章的主要内容包含 5 个方面：数据库的基本概念、常用数据库软件 SQL Server 的使用要点、SQL 语言、ADO. NET 对象模型访问操作数据库的方法及 ASP.NET 数据控件的使用。

4.1 数据库基础

数据库技术可以提高数据的共享性，使多个用户能同时存取数据库中的数据；减少数据的冗余度，提高数据的一致性和完整性；提供数据与应用程序的独立性，从而减少应用程序的维护代价。目前，数据库技术已经成为各种信息系统的核心和基础。

4.1.1 数据库技术相关概念

1. 数据库

数据库(DataBase，DB)是数据的仓库，但它不仅包括数据本身，而且包括相关数据之间的联系。数据库中的数据不只面向某一项特定应用，而是面向多种应用的，可以被多个用户或多个应用程序共享。

2. 数据库系统

数据库系统(DataBase System，DBS)是指在计算机系统中引入数据库后的系统构成，一般由数据库、数据库管理系统和开发工具、应用系统、数据库管理员和用户构成。

3. 数据库管理系统

数据库管理系统(DataBase Management System，DBMS)是位于用户与操作系统之间的一层数据管理软件。数据库在建立、运行和维护时由数据库管理系统统一管理、统一控制。数据库管理系统能让用户定义和操纵数据，并保证数据的安全性、完整性，以及多用户对数据的并发使用和发生故障后的数据库恢复。

4. 数据库应用系统

数据库应用系统(DataBase Application System,DBAS)是由系统开发人员利用数据库系统资源开发出来的,面向某一类实际应用的应用软件系统。

5. 关系数据库

建立在关系模型上的数据库称为关系数据库(Relational DataBase,RDB),它的核心是 DBMS。关系数据库有下面几个重要的概念。

1) 表

关系数据库中所有数据都是以表的形式给出的,这个表就是关系模型中的关系。表 4-1 是一张学生信息表。

表 4-1　学生信息

学号	姓名	性别	出生日期	班级	专业
20141101231	张三	女	1995-2-2	2014DS01	电商
20141101232	李四	男	1994-12-3	2014DS02	电商
20141101233	王五	男	1995-6-7	2014DS02	电商

2) 记录

一个记录(或者说一行)是一组彼此相关的数据集合。记录在关系模型中称为元组。表 4-1 中列出了 3 条记录。

3) 字段

字段是记录中单独的数据子成分。表中的第一列称为一个字段,一个记录是由若干字段组成的,每个字段有自己的名称和数据类型。在关系模型中,字段称为属性。

表 4-1 中学生的属性有 6 个,即学生表有 6 个字段。

4) 主键

键是能唯一标识每个记录的字段或字段组合。表 4-1 中,学号应作为该学生表的主键。

5) 外键

外键又称为外码,它表示该字段不是本表的主键,但它是另一张表格的主键。假如有另外一张成绩表,里面也有学号这个字段,则该字段的值应来自于学生表的主键,但在成绩表中它只能是外键。

6) 视图

视图是通过数据库的命令执行后得到的记录集合,可以把视图看作是表,但不是真正的表,视图会保存在数据库中,其执行效率要比在程序中执行相同的命令语句高。

7) 存储过程

存储过程是在数据库系统中,一组为了完成特定功能的命令语句集,存储在数据库中。它经过第一次编译后,再次调用不需要再编译,用户通过存储过程可完成对数据库的操作,其效率要比在程序中使用相同的命令语句高得多。

4.1.2 SQL 语句

前面讲到在 SQL Server 查询分析器中可以以 SQL 语句的方式操作数据库。那么，什么是 SQL 语句呢？

1. SQL 语言的概述

SQL(Structured Query Language，结构化查询语言)语言结构简洁，功能强大，简单易学。绝大多数关系型数据库系统如 Oracle、Sybase、DB2、Informix、SQL Server、Access 等都支持 SQL 语言作为查询语言。

结构化查询语言 SQL 是一种介于关系代数与关系演算之间的语言，其功能包括查询、操纵、定义和控制 4 个方面，是一个通用的、功能极强的关系型数据库标准语言。在 SQL 语言中不需要告诉 SQL 如何访问数据库，只要告诉 SQL 需要数据库做什么。

SQL 主要分成以下 4 个部分。

(1) 数据定义。这一部分也称为 DDL，用于定义 SQL 模式、基本表、视图和索引。

(2) 数据操纵。这一部分也称为 DML，数据操纵分成数据查询和数据更新两类，其中数据更新又分成插入、删除和修改 3 种操作。

(3) 数据控制。这一部分也称为 DCL，数据控制包括对基本表和视图的授权、完整性规则的描述、事务控制语句等。

(4) 嵌入式 SQL 使用。这部分内容涉及 SQL 语句嵌入在宿主语言程序中的使用规则。

2. SQL 的数据定义

(1) 数据库中的基本数据类型。不同的数据库系统中，数据类型不尽相同。表 4-2 是 SQL Server 的基本数据类型。

表 4-2 SQL Server 数据库的数据类型(部分)

数据类型	描　述	存　储
char(n)	固定长度的字符串。最多 8000 字符	n
varchar(n)	可变长度的字符串。最多 8000 字符	
nvarchar(n)	可变长度的 Unicode 数据。最多 4000 字符	
binary(n)	固定长度的二进制数据。最多 8000 字节	
varbinary(n)	可变长度的二进制数据。最多 8000 字节	
tinyint	允许从 0 到 255 的所有数字	1 字节
smallint	允许从−32768 到 32767 的所有数字	2 字节
int	允许从−2147483648 到 2147483647 的所有数字	4 字节
bigint	允许介于−9223372036854775808 和 9223372036854775807 之间的所有数字	8 字节
decimal(p,s)	固定精度和比例的数字。允许从 $-10^{38}+1$ 到 $10^{38}-1$ 之间的数字。p 参数指示可以存储的最大位数(小数点左侧和右侧)。p 必须是 1～38 之间的值。默认是 18。s 参数指示小数点右侧存储的最大位数。s 必须是 0～p 之间的值。默认是 0	5～17 字节

续表

数据类型	描　　述	存　储
numeric(p,s)	固定精度和比例的数字。允许从 $-10^{38}+1$ 到 $10^{38}-1$ 之间的数字。p 参数指示可以存储的最大位数(小数点左侧和右侧)。p 必须是 1～38 之间的值。默认是 18。s 参数指示小数点右侧存储的最大位数。s 必须是0～p 之间的值。默认是 0	5～17 字节
float(n)	从－1.79E＋308 到 1.79E＋308 的浮动精度数字数据。参数 n 指示该字段保存 4 字节还是 8 字节。float(24)保存 4 字节,而 float(53)保存 8 字节。n 的默认值是 53	4 或 8 字节
real	从－3.40E＋38 到 3.40E＋38 的浮动精度数字数据	4 字节
datetime	从 1753 年 1 月 1 日到 9999 年 12 月 31 日,精度为 3.33ms	8 字节
smalldatetime	从 1900 年 1 月 1 日到 2079 年 6 月 6 日,精度为 1min	4 字节
date	仅存储日期。从 0001 年 1 月 1 日到 9999 年 12 月 31 日	3 字节
time	仅存储时间。精度为 100ns	3～5 字节

(2) 创建数据库。


【基本语法】

```
CREATE DATABASE <数据库名称>
```

【例 4-1】

```
CREATE DATABASE scoremanage
```

【示例说明】

创建了一个名为 scoremanage 的数据库。

(3) 删除数据库。


【基本语法】

```
DROP DATABASE <数据库名称>
```

(4) 创建新表。


【基本语法】

```
CREATE TABLE <表名>(<字段名称 1 ><数据类型> [NOT NULL] [PRIMARY KEY],<字段名称 2 ><数据类型> [not null],...)
```

【例 4-2】 完整的代码请参看资源包中的示例文件.Ex4-2.sql。

【示例说明】

在 scoremanage 数据库中创建了 4 个数据表。

表结构如下。

```
teacher(tid,name,sex,birthdate,department)
student(sno,name,sex,birthdate,class)
```

```
schooltable(crid,term,coursename,tid,class)
score(id,sno,crid,score)
```

这个数据库主要用于学生成绩管理。

(5) 删除表。



【基本语法】

```
DROP TABLE <表名>
```

(6) 在数据表中添加新的字段。



【基本语法】

```
ALTER TABLE <表名> ADD <字段名称><数据类型>
```

【例 4-3】

```
ALTER TABLE student ADD prof nvarchar(50)
```

【示例说明】

在 student 表中新加了一个字段 prof，student 表的结构变为

```
student(sno,name,sex,birthdate,class,prof)
```

(7) 添加主键。



【基本语法】

```
ALTER TABLE <表名> ADD PRIMARY KEY(<字段名>)
```

(8) 删除主键。



【基本语法】

```
ALTER TABLE <表名> DROP PRIMARY KEY(主键字段名)
```

3. SQL 的数据操纵

数据操纵可分为数据查询和数据更新两类。

1) SQL 的数据查询

SQL 中最常用的是从数据库中获取数据。从数据库中获取数据称为数据库查询，查询数据库通过使用 SELECT 语句实现。



【基本语法】

```
SELECT[ALL|DISTINCT] [TOP N] * |<字段列表> FROM <表名 1>[,<表名 2>]
[WHERE 条件表达式][GROUP BY <字段名>[HAVING<条件表达式>]]
[ORDER BY<字段名>[ASC|DESC]]
```

【语法说明】

(1) 其中 ALL 表示所有的记录，默认即为 ALL；DISTINCT 表示重复的记录只选取第一条；TOP N 表示从记录中选择前 N 条；* 表示选择表中所有的字段；＜字段列表＞表示查询指定的字段，字段之间要用英文状态的逗号分开。

(2) FROM 子句用于指定一个或多个表，如果所选的字段来自不同的表，则字段名前应加表名前缀。

(3) WHERE 子句用于限制记录的选择。

(4) GROUP BY 和 HAVING 子句用于分组和分组过滤处理。它能把在指定字段列表中有相同值的记录合并成一条记录。如果在 SELECT 子句中含有 SQL 合计函数，如 SUM 或 COUNT，那么就为每条记录创建摘要值。HAVING 子句用于对 GROUP BY 分组的记录进行条件的筛选。

(5) ORDER BY 子句决定查找出来的记录的排列序列。ASC 代表升序，DESC 代表降序。

【例 4-4】

```
SELECT name,sex,prof,class FROM student WHERE prof = '电子商务'
```

【示例说明】

从学生表(student)检索所有电子商务专业的学生。

实现来自多个关系的查询时，如果要引用不同关系中的同名属性，则在属性名前加关系列，即通过“关系名.属性名”的形式表示，以示区分。在多个关系上的查询可以用连接查询表示，也可用嵌套查询来表示。

【例 4-5】

```
SELECT AVG(score.score) FROM score
WHERE   score.crid IN (SELECT crid FROM schooltable
WHERE coursename = '高等数学' AND class = '2012DS01')
```

【示例说明】

查询 2012DS01 班高等数学这门课程的全班平均分数。

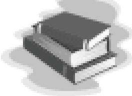
2) SQL 的数据更新

SQL 的数据更新包括数据插入、数据修改和数据删除等操作。

(1) 数据插入。SQL 的数据插入使用 INSERT 语句。

【基本语法】

```
INSERT INTO 表名(字段列表) VALUES (值列表)
```

【语法说明】

值列表中的顺序必须和表中的字段列表一一对应。

【例 4-6】

```
INSERT INTO student(sno,name,sex,birthdate,class,prof)
VALUES('201411011208','张三','男','1993/02/01','2014GS01','工商管理')
```

【示例说明】

向学生表中添加一条新记录。

（2）数据删除。数据删除使用 DELETE 语句。

【基本语法】

```
DELETE FROM <表名> WHERE <条件>
```

【例 4-7】

```
DELETE FROM  student  WHERE sno = '201411011208'
```

【示例说明】

删除学号为 201411011208 的学生信息。

（3）数据修改。数据修改使用 UPDATE 语句。

【基本语法】

```
UPDATE <表名> SET 字段名 1 = 值 1,字段名 2 = 值 2,...WHERE <条件>
```

【例 4-8】

```
UPDATE STUDENT SET class = '2012DS02' WHERE sno = '201211011209'
```

【示例说明】

修改学号为 201211011209 的学生班级为 2012DS02。

4. SQL 的数据控制

目前大多数数据库系统都提供了非常完善的安全机制，一般采用基于角色多级授权安全机制，所有对数据库的操作都需要高一级的授权，任何级别的用户在使用数据库时，除了必须拥有的授权外，还必须提供正确的用户名和口令。在数据库系统中，数据库系统管理员(DBA)负责完成整个系统的管理工作，获得(DBA)授权的用户可以创建数据库、表等而成为这些数据库对象的拥有者。拥有者对自己所拥有的对象有完全控制权，同时拥有者也可以授权其他用户使用所拥有的对象，当然也可以收回授权。

SQL 的数据控制功能是指控制用户对数据的存取权利，语句有两条：授权语句(GRANT)和收回语句(REVOKE)。顾名思义，授权语句是使某个用户具有某些权限，收回语句是收回已授权给用户的权限。用户对数据的存取操作包括增(INSERT)、删(DELETE)、改(UPDATE)和查(SELECT)。只有被授予了某项操作权限的用户才能进行某项操作。

【例 4-9】

```
GRANT ALL ON student TO suser
```

【示例说明】

把对表 student 的所有操作权限授权给用户 suser。

【例 4-10】

```
REVOKE INSERT,UPDATE,DELELTE ON student FROM suser
```

【示例说明】

把用户 suser 对表 student 的更新权收回。

4.2　ADO.NET

ASP.NET 使用的数据库访问技术是 ADO.NET。ASP.NET 通过 ADO.NET 操作数据库的工作方式如图 4-1 所示。本节将重点讲述 ADO.NET 对象模型。



图 4-1　ASP.NET 通过 ADO.NET 访问数据库

4.2.1　ADO.NET 简介

ADO.NET 的名称起源于 ADO(ActiveX Data Objects),这是一个广泛的类组,用于在以往的 Microsoft 技术中访问数据。之所以使用 ADO.NET 名称,是因为 Microsoft 希望表明,这是在.NET 编程环境中优先使用的数据访问接口。

它提供了平台互用性和可伸缩的数据访问。ADO.NET 增强了对非连接编程模式的支持,并支持 RICH XML。由于传送的数据都是 XML 格式的,因此任何能够读取 XML 格式的应用程序都可以进行数据处理。事实上,接收数据的组件不一定就是 ADO.NET 组件,也可以是一个基于 Microsoft Visual Studio 的解决方案,还可以是任何运行在其他平台上的任何应用程序。

ADO.NET 是一组用于和数据源进行交互的面向对象类库。通常情况下,数据源是数据库,但它同样也能够是文本文件、Excel 表格或者 XML 文件。

ADO.NET 允许和不同类型的数据源以及数据库进行交互。然而并没有与此相关的一系列类来完成这样的工作。因为不同的数据源采用不同的协议,所以对于不同的数据源必须采用相应的协议。一些老式的数据源使用 ODBC 协议,许多新的数据源使用 OLEDB 协议,并且现在还不断出现更多的数据源,这些数据源都可以通过.NET 的 ADO.NET 类库来进行连接。

ADO.NET 提供与数据源进行交互的、相关的公共方法,但是对于不同的数据源采用一组不同的类库。这些类库称为 Data Providers,并且通常是以与之交互的协议和数据源的类型来命名的。

4.2.2 ADO.NET 对象模型

图 4-2 是 ADO.NET 对象模型。从图中可以看出，ADO.NET 模型中的对象包括用于建立与数据源之间连接的 Connection 对象；负责存储和执行 SQL 语句的 Command 对象；对数据进行直接的、只读访问的 DataReader 对象；用于创建和操纵 DataSet 类的实例的 DataAdapter 对象。DataSet 是一个支持断开式、分布式数据方案的核心对象。

从图 4-2 还可以看出，以 ADO.NET 访问数据的应用程序，要依赖某种.NET 数据提供程序。.NET 数据提供程序有以下几种。

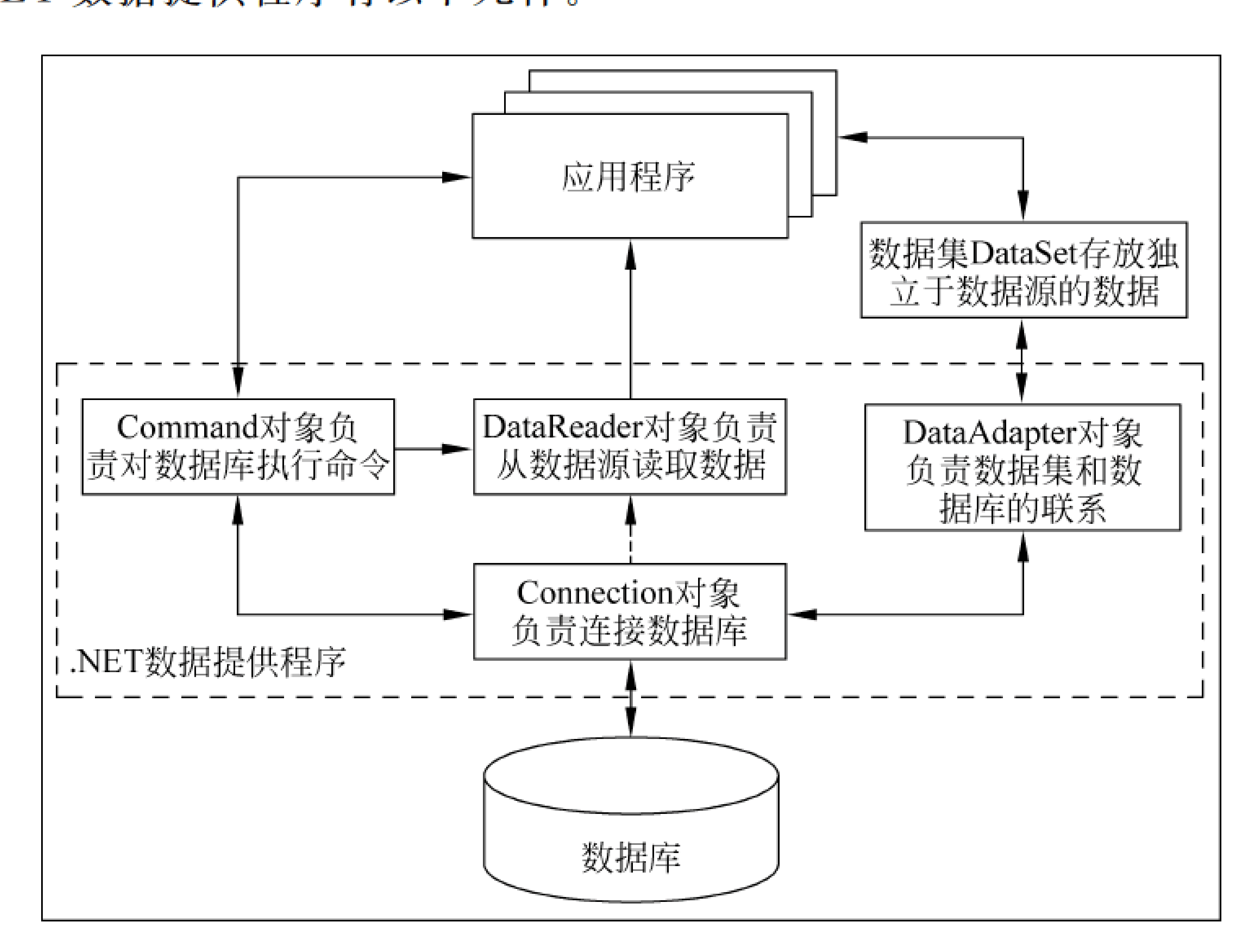


图 4-2 ADO.NET 对象模型

(1) SQL Server 数据提供程序，提供对 SQL Server 7.0 版或以上版本的数据访问，使用 System.Data.SqlClient 空间。

(2) OLE DB.NET 数据提供程序，适用于使用 OLE DB 公开的数据源。使用 System.Data.Oledb 命名空间。

(3) ODBC.NET 数据提供程序，适合于使用 ODBC 公开的数据源。使用 System.Data.Odbc 命名空间。

(4) Oracle.NET 数据提供程序，适用于 Oracle 数据源。使用 System.Data.OracleClient命名空间。

这些.NET 程序为 Connection、Command、DataReader 对象提供独有的类。例如，SQL Server.NET 数据提供程序的 Connection 是通过 SqlConnection 实现的，OleDB.NET 程序提供的 Connection 是通过 OledbConnection 实现的。只有 DataSet 是一个共有的类。

4.2.3 ADO.NET 数据访问对象

由于.NET 数据提供程序不同，ADO.NET 对象的名称也不相同。本章以访问 SQL

Server 数据库为例进行介绍。

使用 SQL Server. NET 数据提供程序前要导入命名空间 System. Data 和 System. Data. SqlClient。在 C# 中使用以下语句进行导入：

```
using System.Data;
using System.Data.SqlClient;
```

1. Connection 对象

Connection 对象用于建立与数据库的连接。Connection 的常用属性如表 4-3 所示，常用方法如表 4-4 所示。

表 4-3　Connection 对象的常用属性

属　性	说　　明
ConnectionString	用来指定连接的字符串
DataSource	用来获取数据源的服务器名或文件名
Database	用来指定要连接的数据库名称
Provide	用来提供数据库驱动程序
State	Connection 对象当前的连接状态

表 4-4　Connection 对象的常用方法

属性	说　　明
Open()	打开连接
Close()	关闭连接
Database	用来指定要连接的数据库名称
Provide	用来提供数据库驱动程序

【例 4-11】 下面是 Connection 对象使用的示例代码，完整代码请参看资源包中的示例文件 Ex4-11. aspx。

```
SqlConnection con = new SqlConnection();
con.ConnectionString = "Data Source = 10.11.57.27;Initial Catalog" +
                       " = scoremanage;User Id = sa;Password = hdwang";
con.Open();
```

【示例说明】

(1) ConnectionString 指定要连接数据库的字符串，对于 SQL Server 而言，Data Source 代表 SQL Server 的服务器名称，Initial Catalog 为数据库的名称，User Id 是连接数据库的用户名，Password 为用户名对应的密码。

(2) 编程时先设置 ConnectionString 的值，然后用 Open 方法打开连接，用完用 Close 关闭。

在实际编程中，连接数据库的 ConnectionString 值并不写在代码中，而是放在 web. config 文件中，web. config 的配置如下，ConS 是连接字符串的名称。

```
<configuration>
<connectionStrings>
<add name = "ConS" connectionString = "Data Source = 10.11.57.27;
Initial Catalog = scoremanage;User Id = sa;Password = hdwang"
providerName = "System.Data.SqlClient" />
</connectionStrings>
</configuration>
```

在程序中用 System. Configuration 命名空间的 ConfigurationManager 读取上述配置

中的 ConS 值。此时例 4-11 中为 ConnectionString 赋值的语句就变为下面的代码。

```
con.ConnectionString = @System.Configuration.ConfigurationManager.
ConnectionStrings["ConS"].ConnectionString;
```

【例 4-12】 完整代码请参看资源包中的示例文件：Ex4-12.aspx。

SqlConnection 对象也可以用下面的代码方式进行声明。

```
String sql = @System.Configuration.ConfigurationManager.
ConnectionStrings["ConS"].ConnectionString;
SqlConnection con = new SqlConnecton(sql);
```

【例 4-13】 完整代码请参看资源包中的示例文件：Ex4-13.aspx。

2. Command 对象

Command 对象表示要对数据库执行 SQL 语句或存储过程。在建立了与数据源的连接后，可以用 Command 对象执行命令并从数据源中返回结果。Command 对象的常用属性如表 4-5 所示，常用方法如表 4-6 所示。

说明如下。

(1) CommandType 默认为 Text 时，CommandText 属性应设为 SQL 语句。

(2) CommandType 设为 StoredProcedure 时，CommandText 属性应设为存储过程名称。

表 4-5　Command 对象的常用属性

属　性	说　　明
CommandType	获取或设置 Command 对象要执行命令的类型
CommandText	获取或设置对数据源执行的 SQL 语句或存储过程名或表名
CommandTimeOut	获取或设置在终止对执行命令的尝试并生成错误之前的等待时间
Connection	获取或设置此 Command 对象使用的 Connection 对象的名称

表 4-6　Command 对象的常用方法

方　法	说　　明
ExecuteNonQuery	执行 SQL 语句并返回受影响的行数
ExecuteScalar	执行查询，并返回查询所返回的结果集中第一行的第一列。忽略其他列或行
ExecuteReader	执行返回数据集的 SELECT 语句
Dispose	释放 Command 对象占有的资源

【例 4-14】 利用 Command 对象执行一条 SQL 插入记录的语句，完整的代码请参看资源包中的示例文件：4-14.aspx。

【示例说明】

(1) 本例先建立连接，然后定义并实例化 Command 对象，通过依次设定 connection、commandtext 属性后，执行 ExecuteNonQuery 方法，完成 SQL 语句。使用完毕后，应用 Dispose 方法释放 Command 对象占有的资源。

(2) 这里所用的 SQL 语句只有更新和删除语句，如果要使用 SELECT 语句，须配合

DataReader 对象。

【例 4-15】 示例功能同例 4-14，但定义 Command 对象的方法不同，完整的代码请参看资源包中的示例文件：4-15.aspx。

【示例说明】

本例先建立连接，设定要执行的 SQL 语句，然后在定义并实例化 Command 对象的同时指定这些参数，这是定义实例化 Command 对象的另一种方式。

【例 4-16】 利用 Command 对象调用存储过程完成 SQL 操作。完整的代码请参看资源包中的示例文件：4-16.aspx。

操作步骤如下。

（1）在 SQL Server 中建立一个带参数的存储过程，用于向 student 表添加记录。

① 启动 SQL Server Management Studio，展开实例名称，依次单击“数据库”→scoremanage 数据库→可编程性→存储过程，在存储过程上右击，选择快捷菜单中的“新建存储过程”命令，如图 4-3 所示。

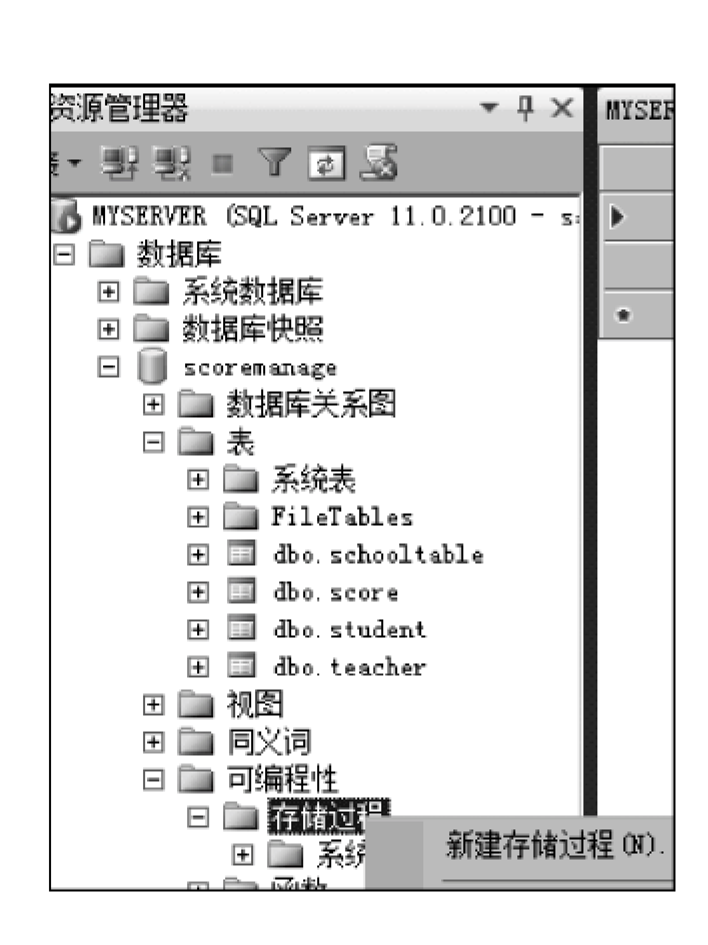


图 4-3　新建存储过程

② 在查询分析器窗口中输入资源包 Ex4-16.sql 的代码，然后单击执行命令，刷新存储过程后就可以看到新建立的 addstudent 存储过程，该存储过程用于向 student 表添加新记录，如图 4-4 所示。

（2）在 Visual Studio 2013 中新建一个页面，编写代码完成调用此存储过程的程序，完整代码请参看资源包中的示例文件：Ex4-16.aspx。

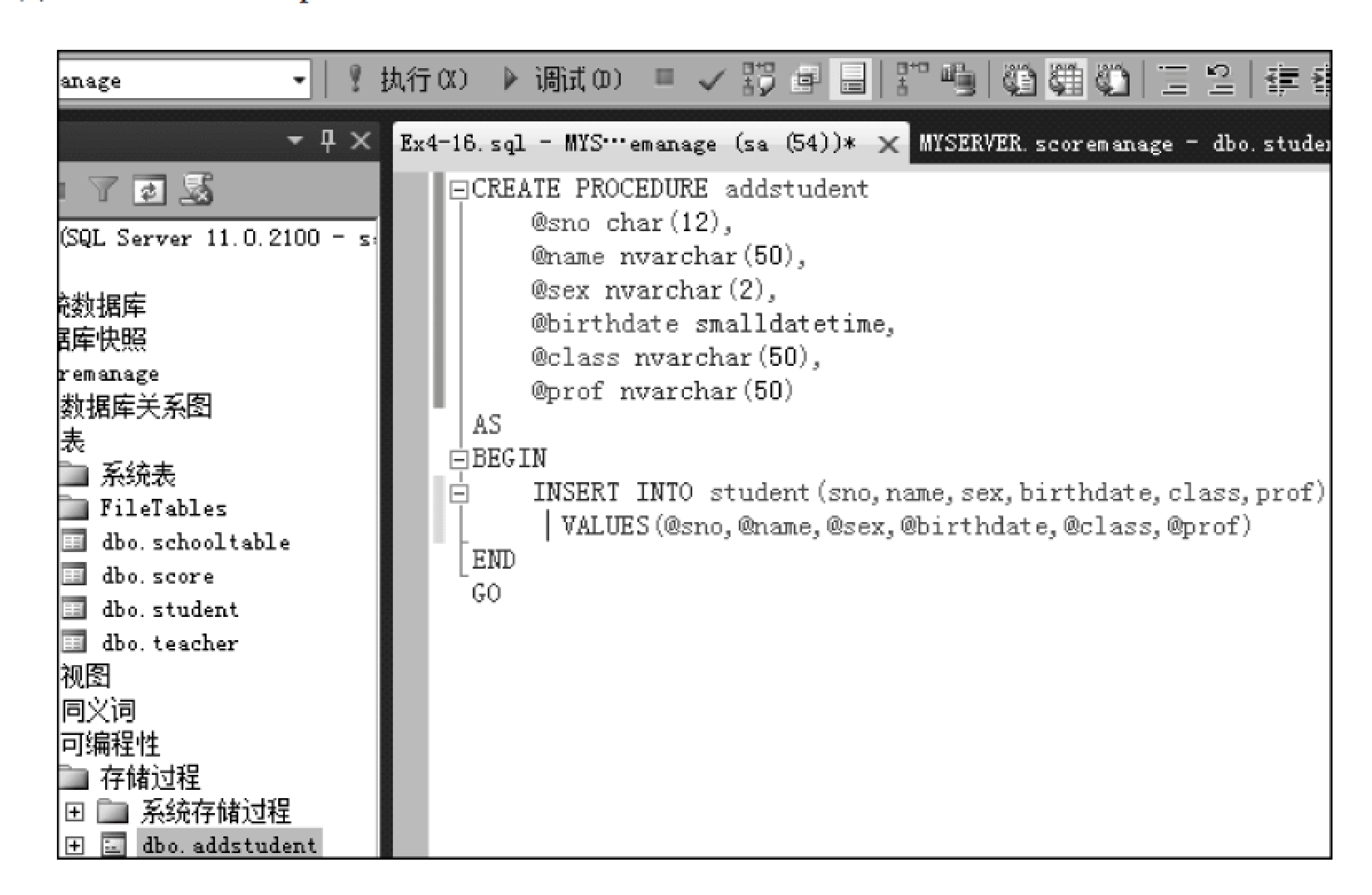


图 4-4　建立的存储过程

【示例说明】

(1) 下面是设定存储过程参数的代码部分:

```
par = new SqlParameter();
par.SqlDbType = SqlDbType.Char;
par.Size = 12;
par.ParameterName = "@sno";
par.Value = "201411011305";
par.Direction = ParameterDirection.Input;
cmd.Parameters.Add(par);
```

(2) SqlParameter 是参数对象,SqlDbType 是参数的数据类型(要与存储过程中参数类型一致),Size 是参数的大小,ParameterName 是参数的名称,Value 是传给参数的实际值,Direction 是参数的传递方向,存储过程中的参数如无 output 关键词,一律为输入方向。将上述参数值设定好后,用 add()方法将参数添加到 Parameters 集合中。

3. DataReader 对象

DataReader 对象用于顺序读取数据,每次以只读的方式读取一条记录,使用 DataReader 对象不但节省资源且效率很高。DataReader 是抽象类,因此不能直接实例化,而是通过 Command 对象的 ExecuteReader 方法返回 DataReader 实例。

DataReader 的常用属性见表 4-7,常用方法见表 4-8。

表 4-7 DataReader 的常用属性

属　性	说　　明
FieldCount	获取当前行的列数
Item	索引器属性,以原始格式获得一列的值
IsClose	获得一个表明数据阅读器有没有关闭的一个值
RecordsAffected	获取执行 SQL 语句所更改、添加或删除的行数

表 4-8 DataReader 的常用方法

方　法	说　　明
Read	使 DataReader 对象前进到下一条记录(如果有)
Close	关闭 DataReader 对象。注意,关闭阅读器对象并不会自动关闭底层连接
Get	用来读取数据集的当前行的某一列的数据
NextResult	当读取批处理 SQL 语句的结果时,使数据读取器前进到下一个结果
GetName	用来获得当前行某一字段的名称
GetValue	用来获得当前行某一字段的值

【例 4-17】 利用 DataReader 对象显示 student 表中的记录,完整的代码请参看资源包中的示例文件:4-17.aspx。

4. DataTable、DataSet 和 DataAdapter 对象

1) DataTable 对象

DataTable 对象是构成 DataSet 最主要的对象,DataTable 对象是由 DataColumns 集合、

DataRows 集合所构成的。DataTable 是内存中的“表”，并不等同于数据库中的原始表。

DataTable 对象的常用属性见表 4-9，常用方法见表 4-10。

DataColumn 对象就是字段对象，是组成 DataTable 的基本单位，其常用属性见表 4-11。

表 4-9　DataTable 对象常用的属性

属　性	说　　明
CaseSensitive	指示表中的字符串比较是否区分大小写
Columns	获取属于该表的列的集合
Constraints	获取由该表维护的约束的集合
DataSet	获取此表所属的 DataSet 以及 DataSet 相关信息
DefaultView	获取可能包括筛选视图或游标位置的表的自定义视图
HasErrors	获取一个值，该值指示该表所属的 DataSet 的任何表的任何行中是否有错误
Primary Key	字段在 DataTable 对象中的功能是否为主键
Rows	获取属于该表的行的集合
TableName	获取或设置 DataTable 的名称

表 4-10　DataTable 对象常用的方法

方　法	说　　明
AcceptChanges()	提交自上次调用 AcceptChanges()以来对该表进行的所有更改
Clear()	清除所有数据的 DataTable
Clone()	复制 DataTable 的结构，包括所有 DataTable 架构和约束
ImportRow(DataRow row)	将 DataRow 复制到 DataTable 中保留任何属性设置以及初始值和当前值
Merge(DataTable table)	将指定的 DataTable 与当前的 DataTable 合并
NewRow()	创建与该表具有相同架构的新 DataRow

表 4-11　DataColumn 对象的常用属性

属　性	说　　明
dataType	列的数据类型，为 Type 类所支持的成员
AllowDbNull	是否允许空值
AutoIncrement	是否自增
AutoIncrementSeed	自增的初始值
AutoIncrementStep	自增的递增值
Caption	DataColumn 的标题
ColumnName	列名
DataType	数据类型，为 Type 类所支持的成员
DefaultValue	默认值
MaxLength	文本类型的最大长度
Ordinal	该列在一个字段集合中的位置
ReadOnly	该列是否只读
Unique	该列值是否唯一

【例 4-18】 完整代码请参看资源包中的示例文件：Ex4-18. aspx。

【示例说明】

（1）createdatatable 函数可以通过代码创建一个 DataTable 对象名称 forum，这个表有 4 个字段，articleid 是一个以 1 为起始，每次增加 1 的自动编号的字段，time 是日期时间类型的字段，title 和 author 都是字符型的字段，同时，该程序为这个表添加了两条记录。

（2）test 过程则是将创建的 DataTable 对象以表格的形式显示在网页上。

2）DataAdapter 对象

DataAdapter 对象用于提供对数据集（或数据表）的填充和对更新的回传任务。它的常用属性如表 4-12 所示，常用方法见表 4-13。

表 4-12　DataAdapter 的常用属性

属　　性	说　　明
AcceptChangesDuringFill	决定在把行复制到 DataTable 中时对行所做的修改是否可以接受
TableMappings	容纳一个集合，该集合提供返回行和数据集之间的主映射
ContinueUpdateOnError	获取或设置当执行的 Update()方法更新数据源时，若发生错误是否继续更新。默认值为 False
DeleteCommand	获取或设置用来从数据源删除数据行的 SQL 命令，属性值必须为 Command 对象，并且此属性只有在调用 Update()方法且从数据源删除数据行时使用
InsertCommand	获取或设置将数据行插入数据源 SQL 命令，使用原则与 DeleteCommand 相同
SelectCommand	获取或设置用来从数据源选取数据行的 SQL 命令
UpdateCommand	获取或设置用来从更新数据源的 SQL 命令

表 4-13　DataAdapter 的常用方法

方　　法	说　　明
Fill()	将 SelectCommand 属性指定的 SQL 命令执行结果所选取的数据行填充到数据集（或数据表）中
Update()	调用 InsertCommand 属性、UpdateCommand 属性或 DeleteCommand 属性指定的 SQL 命令，将数据集（或数据表）对象更新到数据源中

【例 4-19】 完整代码请参看资源包中的示例文件：Ex4-19. aspx。

【示例说明】

（1）本示例演示了通过 DataAdapter 对象填充 DataTable 并对数据源进行更新的操作。

（2）利用 DataAdapter 填充数据表时使用的是 SelectCommand 属性，本示例中新添加了一条记录，程序中并无 InsertCommand 属性的设置，程序是如何将新添加的记录更新到 student 表中的呢？请注意下面这行代码。

```
SqlCommandBuilder cb = new SqlCommandBuilder(adp);
```

这行代码会让程序根据情况自动生成 InsertCommand、UpdateCommand 或 DeleteCommand 属性，然后调用 Update()方法执行这些 SQL 命令。

3）DataSet 对象

DataSet 对象是 ADO. NET 的核心，是离线访问技术的载体，DataAdapter 对象是 DataSet 对象和数据存储之间的桥梁。DataTable 是 DataSet 对象中最重要的部分。DataSet 对象除了 DataTable，还包含主键、外键以及条件约束等信息，它不维持和数据源的连接，其中的数据可以被存取、操作、更新和删除，并保持与数据源中数据一致。

DataSet 对象使用无连接传输模式访问数据源，因此在用户要求访问数据源时，不需要进行连接操作。同时，数据一旦从数据源读入 DataSet 对象，数据源便关闭数据连接，解除数据库的锁定，这样可以避免多个用户对数据源的争夺。

DataSet 对象的内部结构包括 Tables、Relations 和 ExtendedProperties 3 个集合。

(1) Tables 集合。一个 DataSet 包含 0 或多个 DataTable，这些 DataTable 一起构成 Tables 集合。

(2) Relations 集合。一个 DataSet 包含 0 或多个 DataRelation 对象。DataRelation 对象是根据外键值在两个表格间定义的父子关系。这些 DataRelation 一起构成 Relations 集合。

(3) ExtendedProperties 集合。ExtendedProperties 集合用来存储与 DataSet 相关的自定义数据。

DataSet 对象创建的方法可以用下面的代码。

```
DataSet  mydataset = new DataSet("");
```

也可以用已经存在的数据集创建新的数据集对象，如：

```
DataSet mydataset2 = mydataset
```

可以通过程序创建一个带有数据的 DataSet 对象。这些编程方法类似于例 4-18，创建了 DataTable 后将其加入 DataSet 即可。

也可以利用 DataAdapter 对象将数据库中的一个或多个表填充到 DataSet 对象中。

【例 4-20】 完整代码请参看资源包中的示例文件：Ex4-20. aspx。

【示例说明】

本示例演示了通过 DataAdapter 对象将 scoremanage 数据库中的 student 表和 teacher 表填充到 DataSet 对象中。

【例 4-21】 完整代码请参看资源包中的示例文件：Ex4-21. aspx。

【示例说明】

本示例演示了通过 DataAdapter 填充 DataSet、将 DataSet 变动的数据更新到数据源。

4.3　数据库设计

在 4.2 节中，建立了 scoremanage 数据库，里面有 4 张数据表。利用这个数据库学习了 ADO. NET 的对象模型的编程。但这 4 张表的结构有很多地方不符合数据库的设计原则，本节就来对数据库设计进行分析和说明。

4.3.1 数据库设计的4个阶段

数据库设计是对一个给定的应用环境，构造一个最优的数据库模式，并据此建立一个能反映现实世界信息和信息之间的联系，且满足用户对数据及数据加工的要求的数据库及应用系统，使得数据库既能有效、安全、完整地存储大宗数据，又能满足多个用户的信息要求和处理要求。

数据库设计过程是数据库生命周期的一个阶段。数据库生命周期一般包括数据库系统的规划、设计、实现、运行管理和维护、扩充和重构等阶段。一般数据库设计可分为以下4个阶段。

(1) 需求分析。调查与分析设计的对象，对所有可能的数据库用户的数据要求和处理要求进行全面的了解、收集和分析。

(2) 概念模型设计。在需求分析的基础上，构造每个数据库用户的局部视图，然后合并各局部视图，并经优化后形成一个全局的数据库公共视图。这个公共视图就是数据库的概念模型。

这个阶段中需要采用E-R模型进行设计，即画出概念的E-R图。E-R图也称实体-联系图(Entity Relationship Diagram)，它提供了表示实体类型、属性和联系的方法，用来描述现实世界的概念模型。

(3) 逻辑设计。按照一定的准则，将概念模型转换为数据库管理系统能接受的数据模型。

这个阶段需要将E-R图转换成表，实现从E-R模型到关系模型的转换。在操作中，可以用DBMS的DDL进行表的描述。

(4) 物理设计。这是为逻辑数据模型选择一个最合适的物理结构的过程。物理结构主要是指数据库在物理设备上的存储结构和存取方法。

4.3.2 数据库设计的三大范式

数据库的设计范式是符合某一种级别的关系模式的集合。构造数据库必须遵循一定的规则。在关系数据库中，这种规则就是范式。关系数据库中的关系必须满足一定的要求，即满足不同的范式。

目前关系数据库有6种范式：第一范式(1NF)、第二范式(2NF)、第三范式(3NF)、第四范式(4NF)、第五范式(5NF)和第六范式(6NF)。满足最低要求的范式是第一范式(1NF)。在第一范式的基础上进一步满足更多要求的称为第二范式(2NF)，其余范式以此类推。一般说来，数据库只需满足第三范式(3NF)就行了。

在创建一个数据库的过程中，范化是将其转化为一些表的过程，这种方法可以使从数据库得到的结果更加明确。这样可能使数据库产生重复数据，从而导致创建多余的表。范化是在识别数据库中的数据元素、关系以及定义所需的表和各表中的项目这些初始工作之后的一个细化的过程。

1. 第一范式(1NF)

在任何一个关系数据库中，第一范式(1NF)是对关系模式的基本要求，不满足第一范

式(1NF)的数据库就不是关系数据库。

第一范式(1NF)是指数据库表的每一列都是不可分割的基本数据项,同一列中不能有多个值,即实体中的某个属性不能有多个值或者不能有重复的属性。如果出现重复的属性,就可能需要定义一个新的实体,新的实体由重复的属性构成,新实体与原实体之间为一对多关系。简而言之,第一范式就是无重复的列。

2. 第二范式(2NF)

第二范式(2NF)是在第一范式(1NF)的基础上建立起来的,即满足第二范式(2NF)必须先满足第一范式(1NF)。第二范式(2NF)要求数据库表中的每个实例或行必须可以被唯一地区分。为实现区分通常需要为表加上一个列,以存储各个实例的唯一标识。这个唯一属性列被称为主关键字或主键、主码,像学生表中的学号或教师表中的工号就是如此的设计。

第二范式(2NF)要求实体的属性完全依赖于主关键字。完全依赖是指不能存在仅依赖主关键字一部分的属性,如果存在,那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体,新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列,以存储各个实例的唯一标识。简而言之,第二范式就是非主属性非部分依赖于主关键字,通俗地讲,可以称为第三范式就是无重复的行。

3. 第三范式(3NF)

满足第三范式(3NF)必须先满足第二范式(2NF)。简而言之,第三范式(3NF)要求一个数据库表中不包含在其他表中已包含的非主关键字信息。

例如,课程表中有课程的相关信息,如课程名称、课程性质、课时、适用专业等,在设计与课程相关的表时,如学生选课表,就不应该再包含量这些信息,但应在学生选课表中有一列课程编号的字段,需要查询课程信息时,利用课程编号将两个表连接起来进行查看,假如在设计时没有建立课程表,则根据第三范式(3NF)应该构建它;否则就会有大量的数据冗余。简而言之,第三范式就是属性不依赖于其他非主属性。

在实际的数据库设计中,并不一定要全部满足上面 3 个范式。这是因为有时候需要在数据库操作中提高查询的效率,从而牺牲空间来换取时间。

4.3.3 重新设计 scoremanage 数据库

上节创建的 scoremanage 数据库是一个用来进行教学管理的数据库,当时建了 4 个表:

```
teacher(tid,name,sex,birthdate,department)
student(sno,name,sex,birthdate,class,prof)
schooltable(crid,term,coursename,tid,class)
score(id,sno,crid,score)
```

仔细分析,这 4 个表建立的并不符合上面的 3 个范式。像 teacher(教师信息表)中的 department(部门),应该单建立一个部门表 department,用来存储部门的相关信息。与此类似,class(班级)、prof(专业)、course(课程)都应该单建立表,score(成绩)表中不应该有 schooltable(排课表)中的 crid(课表序号)字段,而应该是 course(课程)表中的 crcode(课

程编号)字段。

schooltable(排课表)中的 term 字段也有问题,众所周知,学校一般是按照学年学期进行教学的,所以应单建立一个 term(学年学期表),schooltable 中则应该保留的是 terms 表中的 termid(学年学期编号),同时,score 表也应该加一个 termid 字段。

经过分析,要建立的教学管理数据模型 E-R 图如图 4-5 所示。

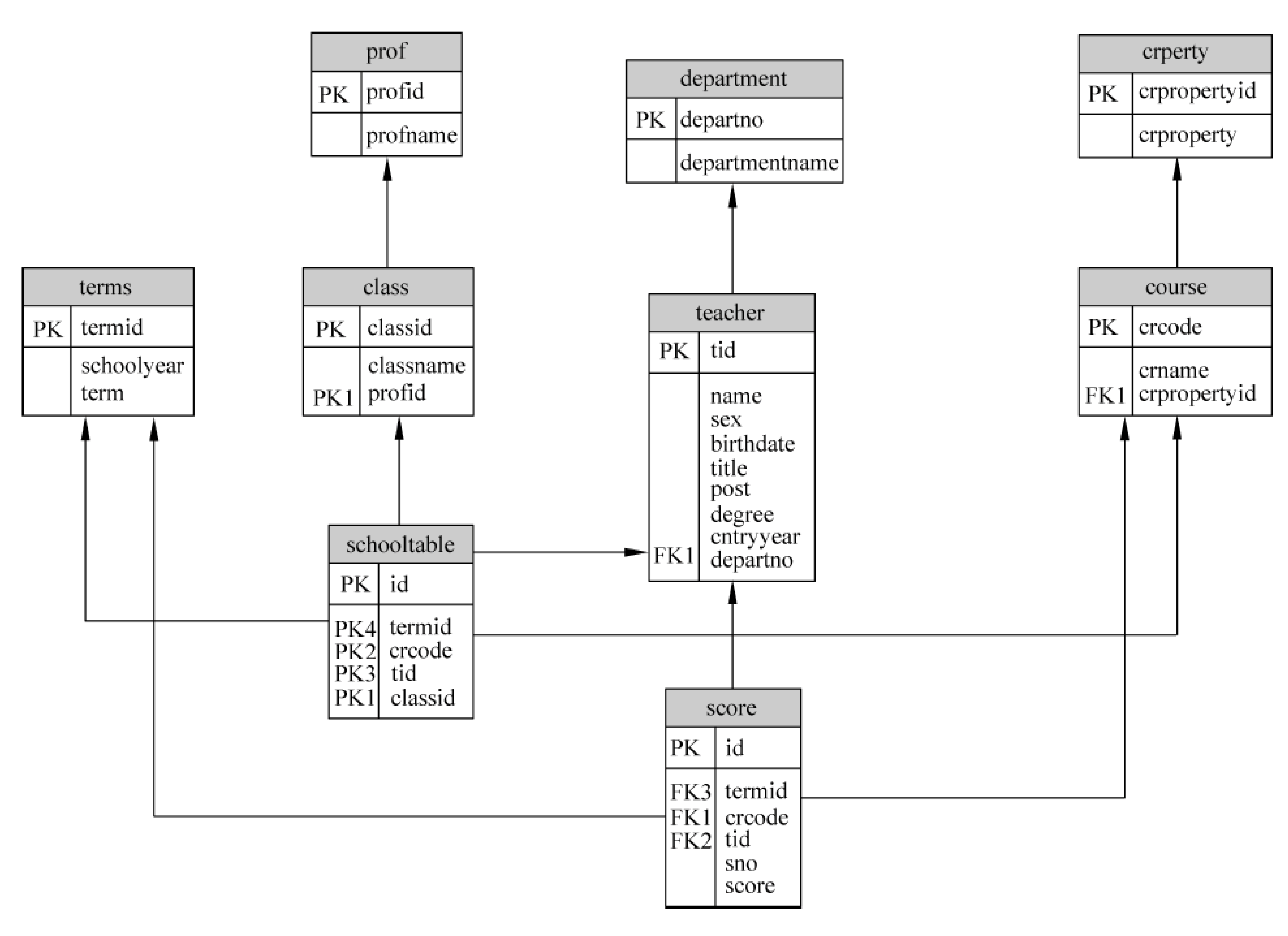


图 4-5 教学管理数据模型 E-R 图

上面的 E-R 图经过转化,就可以设计为下面的 10 张数据表:

student(sno,name,sex,birthdate,classid)
teacher(tid,name,sex,title,post,degree,entryyear,departno)
department(departno,departmentname)
course(crcode,coursename,crpropertyid)
crperty(crpropertyid,crproperty)
class(classid,classname,profid)
prof(profid,profname)
schooltable(id,termed,crcode,tid,classid)
score(id,termed,crcode,tid,sno,score)
terms(termid,schoolyear,term)

表中加实线下划线的字段为该表的主键,加虚线下划线的字段是外键。

【例 4-22】 使用 SQL 语句创建新的 scoremanage 数据库中的数据表。



【小提示】

为了同上一节所使用的示例数据库相区别,这里建立一个新的数据库 Hscoremanage

用来存放这 10 张表。建表的 SQL 语句请参看资源包中的示例文件：Ex4-22. sql。

4.4 ASP.NET 的数据控件

本节讲的数据控件主要是用来显示数据库信息，并能对数据库进行操作的控件。


【小提示】

本节涉及的数据控件，多数都位于控件工具箱中的"数据"选项卡内，请在添加控件的时候注意这一位置。

4.4.1 DropDownList 控件

DropDownList 控件本身就是用来设计下拉列表框功能的普通控件，但这个控件在做网页数据程序时经常会用来显示数据库中的某列数据，以便用户操作，因此在这里先做一下介绍。

众所周知，DropDownList 控件对于选中的项有两个值，一个是 Value(用户看不到)、另一个是 Text(用户看到的)。这对于数据库编程是非常有用的。比如前面介绍的 student 表，显示出来的只有 classid(班级编号)，并不显示 classname(班级名称)，当添加学生记录时，总不能都去输入那些编号吧？

这时，就可以用 DropDownList 控件绑定到 class 表，让每一项的 text 值为 classname，而 value 值则为 classid，这样添加学生记录时只要将选定项的 value 值作为 classid 字段的对应值去操作就可以了。

那么 DropDownList 控件是如何能够做到上面所说的功能的呢？

DropDownList 控件有个 DataSource 属性，如果将 DataSource 设置为已经填充好的 class 数据表，然后分别指定 DropDownList 的 DataTextField 属性为 classname，DataValueField 为 classid 就可以了。

【例 4-23】 DropDownList 控件显示班级信息，程序请参看资源包中的示例文件：Ex4-23. aspx。

【示例说明】

这行代码是以对话框的形式在网页中显示 DropDownList 控件选定项的 Text 值和 Value 值。Alert 是 Javascript 脚本，用于给出提示信息(注意信息内容用，分隔)。

```
Response.Write("<script>alert('您选中的班级是:" + dpclass.SelectedItem.Text + ",班级
编号是:" + dpclass.SelectedValue + "');</script>");
```

【例 4-24】 编程实现添加学生信息的功能，程序请参看资源包中的示例文件：Ex4-24. aspx。

【示例说明】

(1) 本示例利用 ASP.NET 调用 SQL Server 的存储过程 addnewstudent(生成存储过程的程序清单请见资源包中的 Ex4-24. sql)完成添加学生信息的操作。

(2) 这个存储过程较复杂,参数中有个 output,用于反馈插入记录的情况。在这个存储过程中,需要先判定有没有添加的@sno(学号)信息。如有,则说明该学号的学生已经存在,这时不进行插入操作;如果没有,才进行插入操作。本例采用了动态 SQL 的执行方法,请仔细分析和体会,注意字符串标志的,符号在动态 SQL 中如何进行处理。

(3) 在添加记录的处理过程中,需要判断学生信息有没有填写,没有填写的信息应给出提示,让操作者补全再进行添加,添加成功或不成功则给出相应的提示。

4.4.2 SQLDataSource 控件

SQLDataSource 是访问 SQL Server 数据库的数据源控件,使用它可以直接绑定 SQL Server 数据库中的数据,通过与其他数据绑定控件一起使用,可以检索、显示、编辑和排序数据,完成这些操作不必编写代码或只需写少量代码。

【例 4-25】 使用 SQLDataSource 数据源控件和 GridView 控件显示 Student 表。

(1) 在 web.config 配置文件中新建一个名为 ConSH 的 ConnectionString,用于连接 Hscoremanage 数据库。

(2) 新建一个 C#的 Web 窗体页面,命名为 Ex4-25.aspx,然后从控件工具箱的"数据"选项卡中将 SQLDataSource 控件加入 Ex4-25.aspx 页面。

(3) 单击 Ex4-25.aspx 页面上的 SQLDataSource 控件右侧的箭头,单击弹出的"配置数据源",出现选择数据连接的窗口,如图 4-6 所示。在这里选择刚才建立的连接 ConSH。

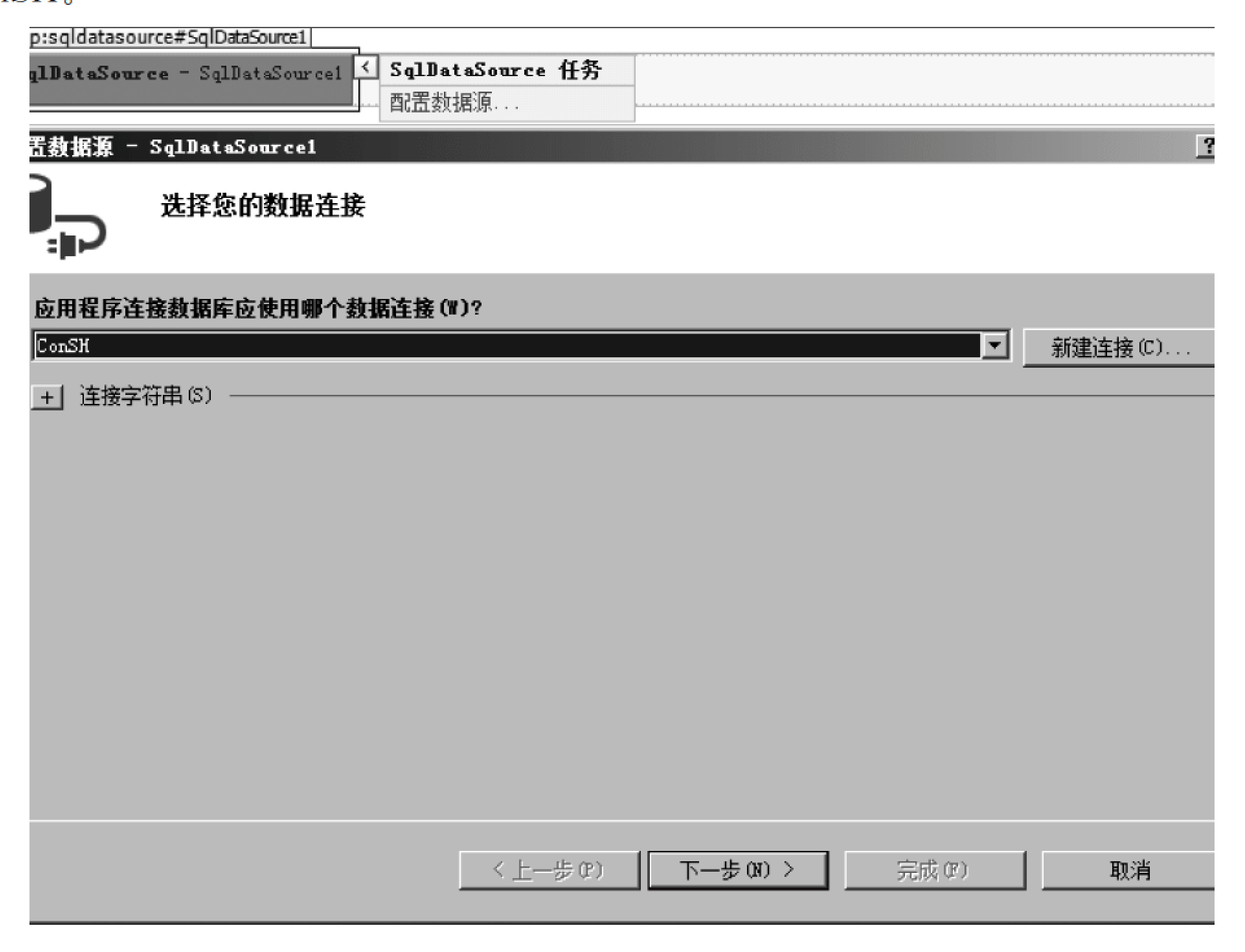


图 4-6 "配置数据源"选择数据连接

(4) 下一步选择要显示的表或自定义 SQL 语句用于配置 Select 语句，在这里选择 student 表的所有字段（在 * 复选框中上打钩），如图 4-7 所示。



图 4-7　“配置数据源”配置 Select 语句

(5) 下一步是测试查询效果，单击“测试查询”按钮，会显示 student 表的检索结果，如图 4-8 所示。此时单击“完成”按钮。



图 4-8　测试查询效果

(6) 将控件工具箱“数据”选项卡中的 GridView 控件添加到资源包中 Ex4-25.aspx 页面中，并为其 DataSourceID 属性选择 SqlDataSource1（这是刚才加到页面上的

SQLDataSource 控件的默认名称)，如果在加入时更改了 SQLDataSource 控件的名称，请选择更改后的名称。

(7) 在浏览器中浏览资源包中的 Ex4-25.aspx 页面，就可以看到数据显示在页面中，如图 4-9 所示。



【小提示】

图 4-9 中显示出来的班级信息是班级编号(classid)，每个字段名称都不是中文，这些都很不友好，想想如何显示成图 4-10 所示的效果呢?

sno	name	sex	classid	birthdate
47200201316	杨富强	男	25	
47200201609	吴寅秋	女	25	
47200202102	周宇菲	男	81	
47200202113	李鹏	男	30	
47200202206	张薇	男	14	
47200202233	张宏峰	男	30	

图 4-9 浏览资源包中的 EX4-25.aspx 效果

学号	姓名	性别	班级	出生日期
47200201316	杨富强	男	2003普营	
47200201609	吴寅秋	女	2003普营	
47200202102	周宇菲	男	2004药剂(3)	
47200202113	李鹏	男	2003中药暑期	
47200202206	张薇	男	2004中药(2)	

图 4-10 学生信息显示效果

【例 4-26】 完成图 4-10 的显示效果设计，答案请参看资源包中的示例文件：Ex4-26.aspx 页面。

【示例说明】

GridView 是一个以表格形式展示数据的控件，其 DataSourceID 绑定数据源控件后，会自动展示数据源。

SQLDataSource 控件和 GridView 控件配合不仅可以显示数据，还可以对数据进行更新和删除。

【例 4-27】 使用 SQLDataSource 数据源控件和 GridView 控件对 Student 表中的记录进行删除和修改。

(1) 重复例 4-25 中的步骤(1)～(6)，建立一个能显示 student 表的 Ex4-27.aspx 页面(见资源包)。

(2) 选中 GridView 控件，在属性窗口中将其 AutoGenerateDeleteButton 的属性改为 True，将 AutoGenerateEditButton 的属性改为 True。这两个属性的作用是分别显示删除和编辑两个链接按钮，如图 4-11 所示。

(3) 选中 SQLDataSource 控件，在属性窗口中找到 DeleteQuery 属性，单击该属性右侧的按钮，弹出命令和参数编辑器窗口。在参数部分单击“添加参数”，将名称改为 sno，然后在“参数源”下拉列表框中选择 QueryString，DefaultValue 的值可以不用填，在“DELETE 命令”框中输入 delete from student where sno=@sno，如图 4-12 所示，完成后单击“确定”按钮。

(4) 同时，利用命令和参数编辑器编辑 UpdateQuery 属性，如图 4-13 所示。

(5) 运行资源包中 Ex4-27.aspx 页面，试着对记录进行添加和删除的操作。

【示例说明】

(1) 设计的 student 表一定要有主键(这张表的主键是 sno)；否则不能进行删除和更

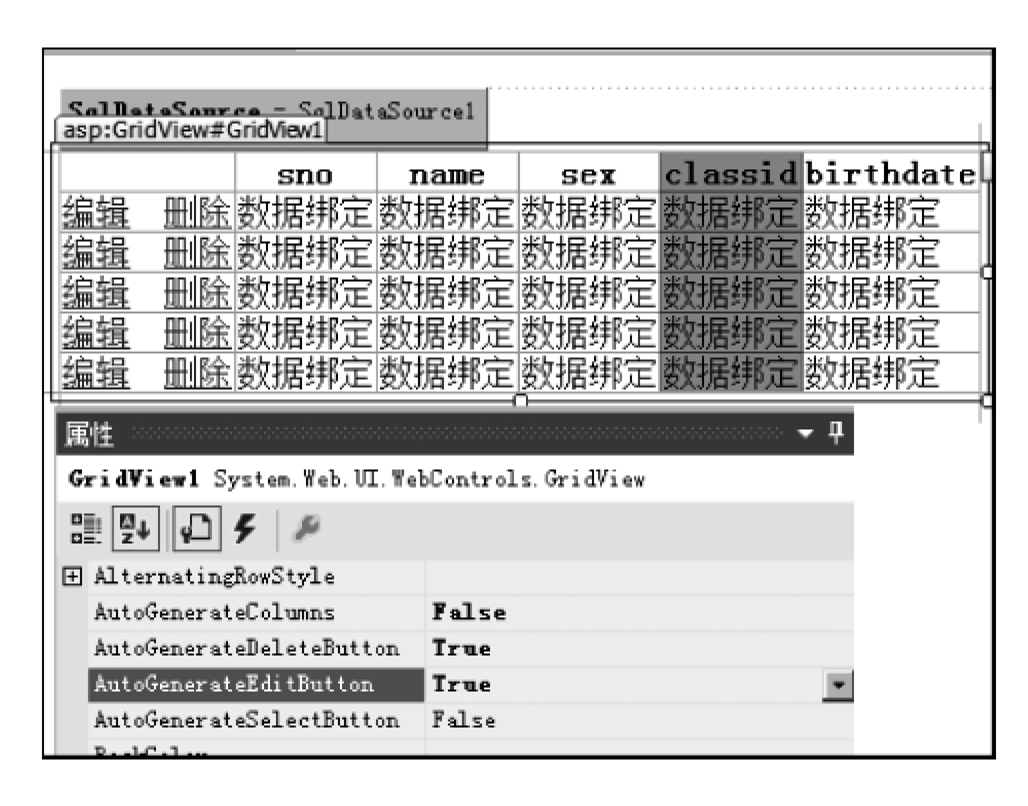


图 4-11　允许 GridView 控件显示删除和更新链接按钮

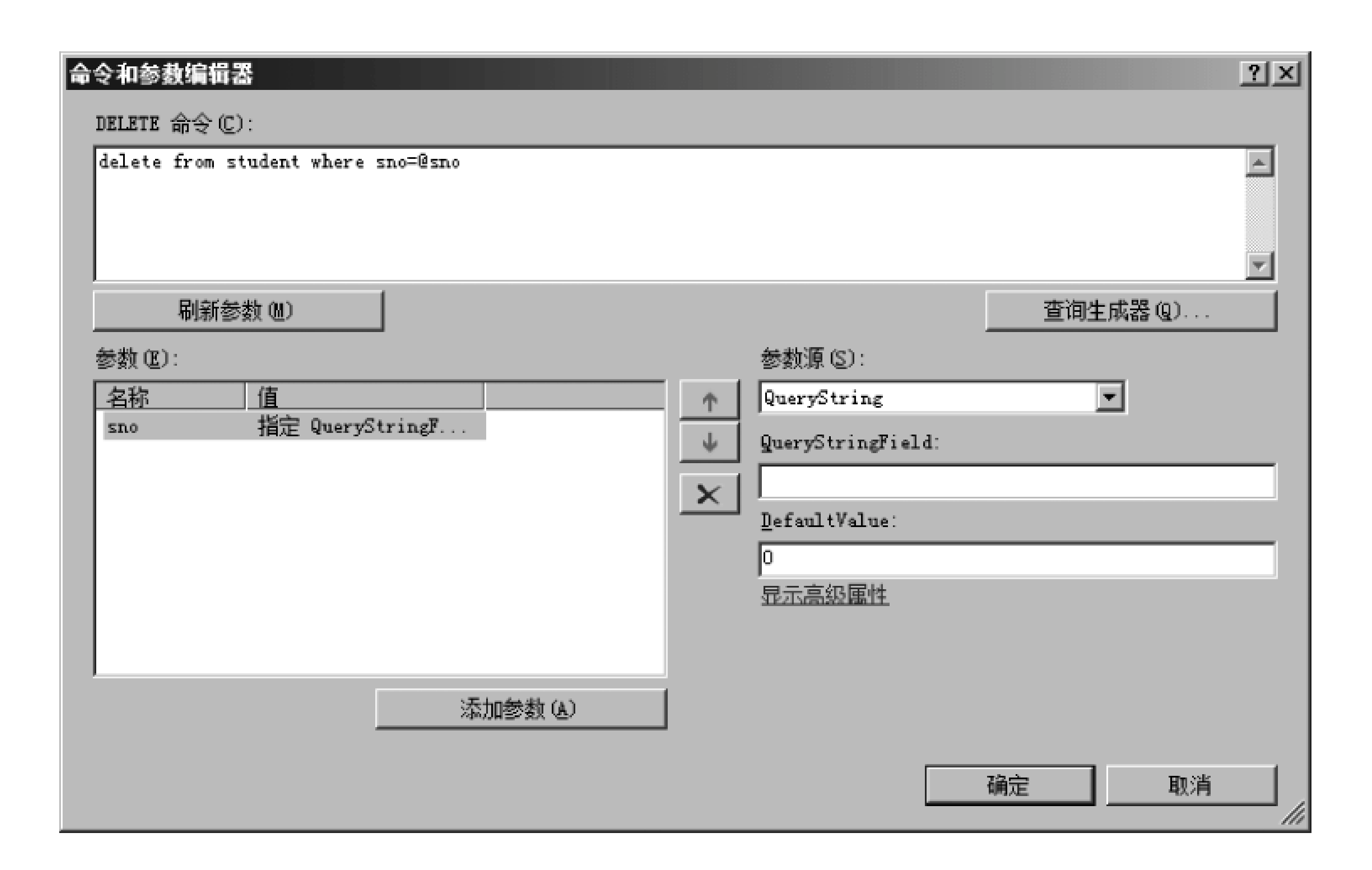


图 4-12　DeleteQuery 命令和参数编辑器



图 4-13　UpdateQuery 命令和参数编辑器

新操作。

(2) 当将 SQLDataSource 控件绑定到数据表时,SQLDataSource 控件会自动识别主键,这样 GridView 控件绑定 SQLDataSource 控件后,GridView 控件的 Datakeynames 会自动设置为主键(sno)。

(3) 当设置了 GridView 控件的 AutoGenerateDeleteButton 为 True 后,GridView 会显示删除列,当用户单击某一行的删除按钮后,GridView 会自动将要删除的列的主键以 QueryString[主键值]的形式传递给 SQLDataSource 控件,由于设置了 SQLDataSource 的 DeleteQuery 属性,SQLDataSource 能够自动将主键值以 QueryString 的方式接收过来,赋给参数 sno,然后调用 Delete 语句,删除指定的主键对应的记录。

(4) DeleteQuery 中的 Delete 语句中的@sno 代表的是接收过来的参数的实际值。

(5) 更新操作原理类似于删除操作,只不过更新操作要传递多个参数。

4.4.3 GridView 控件

GridView 控件用于显示数据,并通过一定的设计允许对数据源进行更新、删除等操作。GridView 控件还支持分页显示。

它绑定数据的方式有两种。

(1) 通过该控件的 DataSourceID 属性进行数据绑定。

(2) 通过该控件的 DataSource 属性进行数据绑定。

使用 DataSourceID 属性,一定要绑定到数据源控件,这种方式较为简单,几乎不用写代码,但功能相对来说比较有限。使用 DataSource 属性进行绑定,可以绑定到 ADO.NET 的数据集或数据表,但这种方式对所有的操作都要编写代码来实现。

在讲 SQLDataSource 控件中已经接触到了 GridView 控件,下面通过编程实例来深入学习 GridView 控件。

1. 分页技术

运行例 4-25 时大家会发现,如果 student 表中的记录很多时,删除操作会显得很慢,这主要是由于 GridView 一次显示了 student 表中的所有记录,用户既不得翻看,而且对数据库的操作效率也很低。通常情况下,会对 GridView 进行分页显示,即每次只显示其中的一部分记录,然后通过翻页的方式浏览其他记录。

当设置了 GridView 控件的分页属性后就可以进行分页显示了。这些属性见表 4-14。

表 4-14 GridView 控件的分页相关属性

属　性	说　明
AllowPaging	当为 True 时允许分页显示,默认为 False
PageSize	每页显示的记录数目,默认是 10
Mode	分页的显示方式
FirstPageText	分页时第一页链接的显示文字
LastPageText	分页时最后一页链接的显示文字

续表

属 性	说 明
NextPageImageUrl	分页时下一页链接的显示文字
PreviousPageImageUrl	分页时上一页链接的显示文字
position	分页链接显示的位置
FirstPageImageUrl	当以图形方式显示分页文字时，第一页的显示图片
LastPageImageUrl	当以图形方式显示分页文字时，最后一页的显示图片
FirstPageImageUrl	当以图形方式显示分页文字时，第一页的显示图片
NextPageImageUrl	当以图形方式显示分页文字时，下一页的显示图片
PreviousPageImageUrl	当以图形方式显示分页文字时，上一页的显示图片
PageIndex	确定当前所显示页面的索引或以编程方式更改显示的页面

【例 4-28】 在例 4-27 的基础上对 student 表进行分页显示，每页显示 5 条记录。

【示例说明】

(1) 将 GridView 的 AllowPaging 改为 True。

(2) 将 PageSize 改为 5。

【例 4-29】 在例 4-28 的基础上对分页样式进行设定。请参见 Ex4-29.aspx 页面(见资源包)。

2. DataSource 方式绑定数据

【例 4-30】 DataSource 绑定方式绑定到 student 表，并分页显示，请参见 Ex4-30.aspx 页面(见资源包)。

【示例说明】

(1) 在新建的页面上只需添加一个 GridView 控件，并设置其 AllowPaging 为 True。

(2) Bind()是自定义的绑定 GridView 到数据表 student 的过程。

(3) GridView1_PageIndexChanging 是翻页时发生的事件，GridView1.PageIndex = e.NewPageIndex；这句代码是将 GridView1 的当前页设置为用户选择的页面。通过编程的方式分页时一定要重新调用 bind()过程。

3. 自动生成列和绑定列

在前面的示例中，当 GridView 控件绑定到数据源后，运行时自动显示数据源的数据，这是由于在默认状态下，GridView 控件的 AutoGenerateColumns 属性为 True，这个属性决定 GridView 控件绑定到数据源时是否自动产生列。如果把它改为 False，GridView 就不会自动产生列了。

使用自动生成列的功能虽然方便，但也有很多问题，比如显示的是数据库中数据的原始格式，有时可能需要改变数据的显示方式，还有就是有些字段不想显示，需要隐藏，这样就不能使用自动生成列了，而是可以采用定制列的方式来进行数据的显示。

【例 4-31】 在例 4-30 的基础上对 GridView 进行编辑列，定制自己需要的列。请参见 Ex4-31.aspx 页面(见资源包)。

(1) 重复例 4-30 的步骤，将 GridView 绑定到 student 表，然后在“设计”视图中单击

GridView 控件右侧的箭头，在 GridView 任务中选择“编辑列”，进入定制列设置框，如图 4-14 所示。

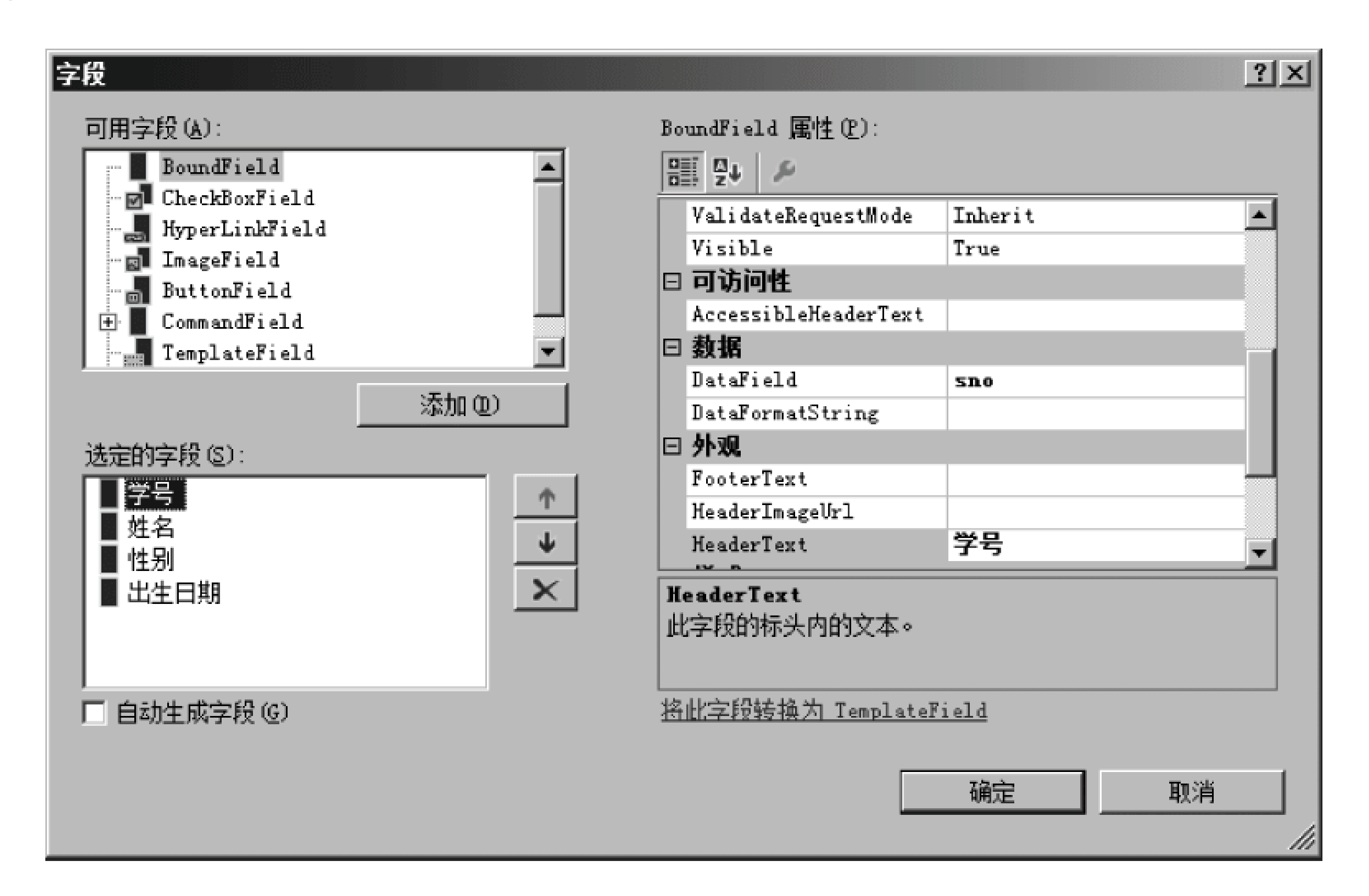


图 4-14 GridView 绑定列窗口

(2) 将左下方的“自动生成字段”前面的钩去掉，在可用字段上选择 BoundField，然后单击“添加”按钮，此时在“选定的字段”中会出现一个 boundfield 字段名称，选中它，在右侧的 BoundField 属性中将 DataField 属性设置为 sno，将 HeaderText 属性设置为“学号”，依次添加 name(姓名)、sex(性别)、birthdate(出生日期)，然后单击“确定”按钮。

(3) 运行 Ex4-31. aspx 页面，观察结果。

【示例说明】

(1) DataField 是数据源中的字段名称，当 GridView 控件通过设置 DataSource 绑定到数据源时，该绑定列就会自动显示该字段对应的记录值。如果数据源中没有这个字段名称，则会发生错误。

(2) HeaderText 是 GridView 控件中字段标头的文本。即显示的字段名称可以不是数据源中的真实字段名称，类似于 DataColumn 中的 caption 属性。

(3) 通过这种绑定列的方式，可以隐藏不需要看到的列，同时，利用 DataFormatString 属性，还可以对字段显示的方式进行设置。

【例 4-32】 用 DataFormatString 设置出生日期的格式，请参见 Ex4-32. aspx 页面(见资源包)。

【示例说明】

注意以下设置：

```
DataFormatString:{0:yyyy-MM-dd}
```

4. 使用模板列

GridView 控件可以使用 TemplateField(模板字段)来扩展功能。TemplateField 允许每一列定义一个完全定制的模板，在模板中可以加入控件标记、HTML 元素以及数据

表达式等，即可以按照自己的需要来布置 GridView 控件，这使得 GridView 控件具有了强大的功能。

【例 4-33】 使用模板列功能完成学生信息的显示、更新和删除功能。请参见 Ex4-33.aspx 页面（见资源包）。

（1）新建 C＃的网站 Web 窗体，重复例 4-31 的步骤，将 classid（班级编号）字段也添加进绑定列。

（2）在 GridView 的绑定列窗口（图 4-14），选择 name 绑定列，单击右上方的“将此字段转换为 TemplateField”链接，此时 name 列就转化为模板列了。按照此方式，将除 sno 字段外的其他字段全部转化为模板列。

（3）当退出绑定列窗口后，再次单击 GridView 右上方的箭头，在 GridView 任务中选择“编辑模板”，GridView 任务就进入了模板编辑模式，如图 4-15 所示。

（4）单击图 4-15 所示的“显示”下拉列表框按钮，可以看到模板列中显示的所有模板。如图 4-16 所示。GridView 各模板的含义见表 4-15。在使用模板列后，当看到数据显示的状态时，GridView 会启用 ItemTemplate 模板，默认情况下，每个模板中是一个 Label 控件，只用来显示绑定的字段所对应的记录值，而当 GridView 进入编辑模式后，就会启用 EditItemTemplate 模板，默认情况下每个模板中是一个文本框控件，用户在此可以修改记录。



图 4-15　GridView 的编辑模板

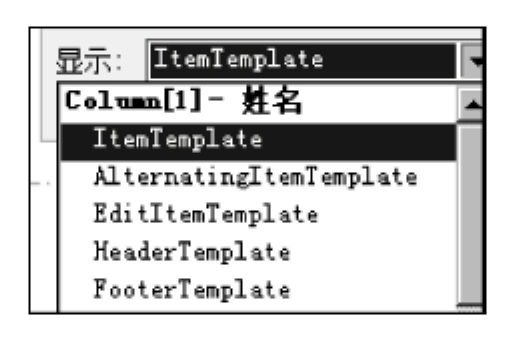


图 4-16　模板列内容

表 4-15　GridView 控件的模板

模 板 名 称	说　　明
ALternatingItemTemplate	间隔行模板
EditItemTemlpate	编辑行模板
FooterTemplate	脚注模板
HeaderTemplate	标题模板
ItemTemplate	每个显示行模板

（5）如图 4-17 所示，选择姓名字段的 EditItemTemplate 模板，如图 4-18 所示，里面有一个文本框控件，选中它，将其 ID 改为 txtname，如图 4-18 所示。



图 4-17　EditItemTemplate 模板中的文本框控件

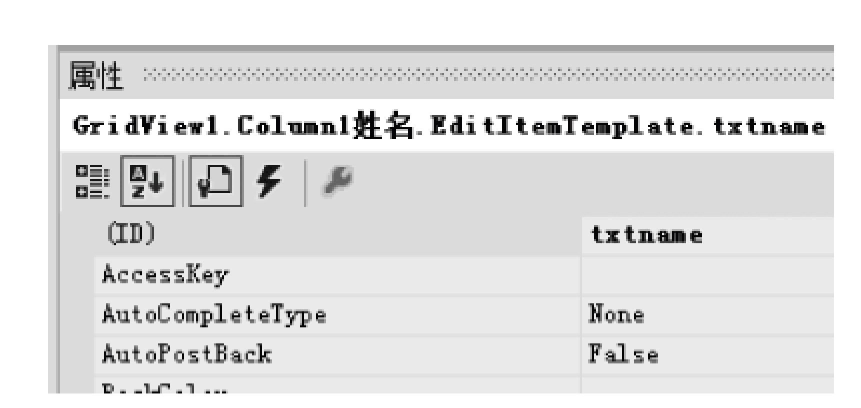


图 4-18　修改模板中控件属性

(6) 按照步骤(5)的操作方法,依次将绑定到 sex、birthdate、classid 字段的 EditItemTemplate 模板中的文本框 ID 改为 txtsex、txtbirthdate、txtclassid。然后退出模板编辑模式。

(7) 添加 GridView 控件的 DataKeyNames 属性值 sno,然后单击 GridView 任务,进入编辑列,选择 CommandField 字段,添加"编辑""更新""取消"和"删除"两列,如图 4-19 所示。然后单击"确定"按钮。

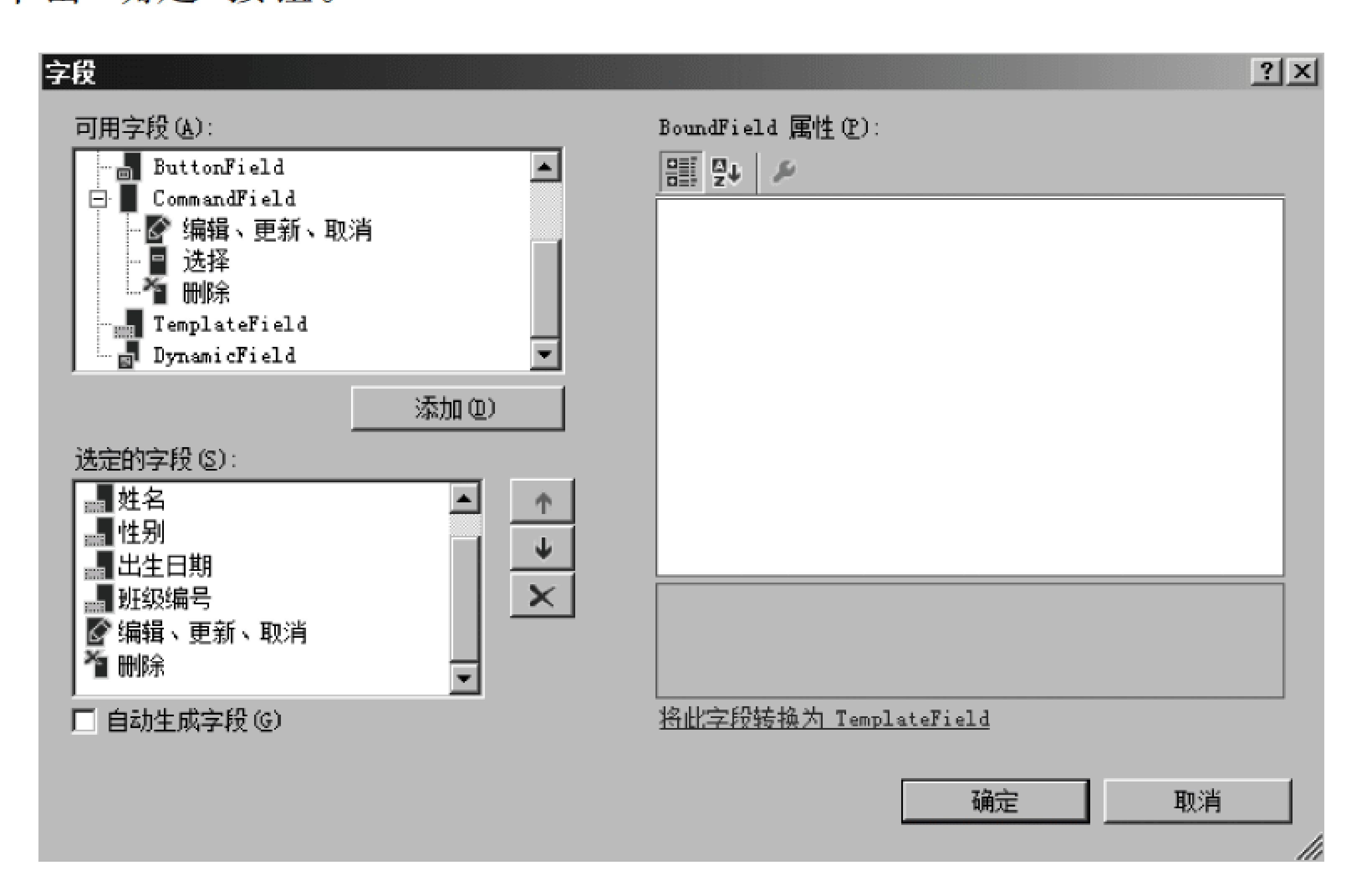


图 4-19 添加 CommandField 字段

(8) 进入后台代码窗口,依次为 GridView1_RowEditing、GridView1_RowDeleting、GridView1_RowUpdating 事件编写代码。具体代码见 Ex4-33.aspx(见资源包)。

(9) 保存文件,运行页面,测试更新、删除的操作,观察结果。

【示例说明】

(1) DataKeyNames 属性是指定主键,这样,当删除或编辑 GridView 的某一行时,通过 GridView1.DataKeys[e.RowIndex].Value.ToString()就可以得到该行对应的 sno 值了。

(2) GridView1_RowEditing 是进入编辑状态的事件,GridView1.EditIndex = e.NewEditIndex 将当前行设为编辑状态,即该行显示为 EditItemTemplate 模板。

(3) GridView1_RowDeleting 是记录删除事件,利用 DataKeys 属性得到要删除的 sno 值,利用 Command 对象执行 Delete 语句,从而删除数据源中的该条记录。

(4) GridView1_RowUpdating 是更新记录的事件。该事件的代码是在处于编辑状态中的 EditItemTemplate 模板中依次找出记录值所对应的文本框中的内容,然后利用 Update 语句执行更新操作。下面的代码是在模板行中查找姓名所对应的文本框,然后得到文本框中修改的字段值(步骤(5)、(6)修改 EditItemTemplate 中文本框的名称就是为此)。

```
GridViewRow myrow = GridView1.Rows[e.RowIndex];
TextBox mybox;
```

```
mybox = (TextBox)myrow.FindControl("txtname");
String name = mybox.Text;
```

【例 4-34】 在模板中使用 DropDownList 控件。请参见 Ex4-34.aspx 页面(见资源包)。

【示例说明】

(1) 在例 4-33 中,性别、班级编号字段在更新时需要输入,这显然不符合用户需求,应该提供下拉列表框,让用户通过选择的方式进行输入。操作步骤同例 4-33 基本一样,需要在将这两个字段的 EditItemTemplate 模板列中的文本框删除,换为 DropDownList 控件。

(2) 性别的 DropDownList 可以直接添加 Items,对于班级编号的 DropDownList,需要绑定到 class 表。这个绑定过程可以利用一个 SqlDataSource 控件完成。

(3) 切换到源视图,分别对 dpsex 和 dpclass 控件添加以下代码:

```
selectedValue = '<% #Bind("sex") %>'
selectedValue = '<% #Bind("classid") %>'
```

这两行代码的作用是在运行时将 dpsex 和 dpclass 与数据源对应的值绑定,以显示数据源中的字段对应值。

(4) 在 RowUpdating 事件中,注意取控件值的写法有所变化,因为性别和班级编号已经变成了 DropDownList 控件。

运行结果类似于图 4-20。

学号	姓名	性别	出生日期	班级编号		
2	4	男	1988-09-07	3	编辑	删除
2555	544	男	1995-02-01	2	编辑	删除
35	4	男	1987-12-09	2008中药1班	更新 取消	
4710012	myname2	男	2007-10-05	8	编辑	删除

图 4-20　绑定了 DropDownList 控件的 GridView 运行界面

5. 转到详细页

GridView 控件是以表格的形式显示数据,有时还会根据情况隐藏某些字段,可能需要转到一个详细页来显示完整的记录。转跳的方法是在 GridView 中使用 HyperLinkField 列。

【例 4-35】 在模板中使用 HyperLinkField。本示例有 Ex4-35.aspx 和 Ex4-35-showstudent.aspx 两个页面(见资源包)。

【示例说明】

(1) Ex4-35.aspx 页中的 GridView 控件编辑列时,姓名采用 HyperLinkField 列,设置其属性如图 4-21 所示。DataNavigateUrlFormatString 属性中设置的 Ex4-35-showstudent.aspx sno={0}是用于跳转到 Ex4-35-showstudent.aspx 页面,并将对应行的 sno 值传递过去。

(2) Ex4-35-showstudent.aspx 页是显示详细信息页,设计这个页面时在上面添加一个 DetailsView 控件,然后编写代码填充这个控件就可以了。String s = Request

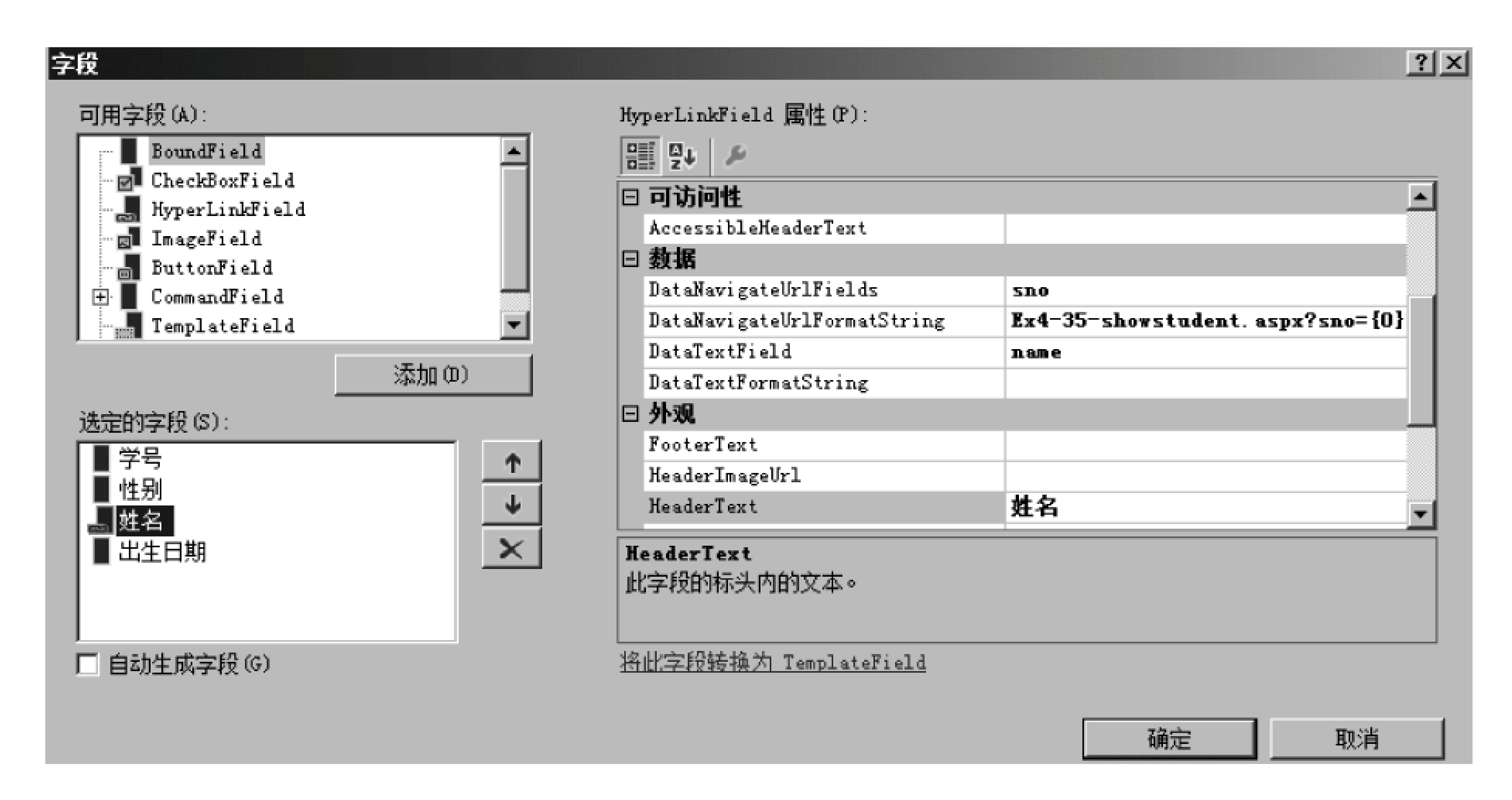


图 4-21 设置 HyperLinkField 属性的值

["sno"].ToString();这行代码是用于接收从 Ex4-35.aspx 页面(见资源包)传递的 sno 值。

4.4.4 DetailsView 控件

该控件能以表格形式详细显示每一行数据中各个数据字段的值。其表格只包含两个数据列;一个数据列逐行显示数据列名;另一个数据列显示与对应列名相关的详细字段信息。

DetailsView 控件支持数据源控件,内嵌了对于显示、编辑、插入或删除的支持。

在例 4-35 中已经接触到了 DetailsView 控件显示记录的编程方法,下面用这个控件编辑 teacher 表。

【例 4-36】 采用 DetailsView 控件编辑 teacher 表,详见 Ex4-36.aspx 页面(见资源包)。

(1) 新建 Web 窗体页面,添加 SqlDataSource 控件和 DetailsView 控件。

(2) 在配置 SqlDataSource 的数据源为表 teacher,并分别设置 INSERTQUERY、UPDATEQUERY、DELETEQUERY 属性。

(3) 将 DetailsView 控件的 DataSourceID 属性设为 SqlDataSource1,单击 DetailsView 控件右上方的箭头,在 DetailsView 任务中选中“启用分页”“启用插入”“启用编辑”和“启用删除”复选框。

(4) 运行该页面,观察现象。

4.4.5 FormView 控件

该控件与 DetailsView 控件类似,用于一次从其关联的数据源呈现一条记录,同时可选择显示分页按钮,以在记录之间进行导航。但 FormView 控件要求用户使用模板定义每项的列表呈现,因此具有更大的灵活性。

【例 4-37】 采用 SqlDataSource 和 FormView 控件编辑 teacher 表，详见 Ex4-37.aspx 页面(见资源包)。

4.4.6 Repeater 控件

Repeater 控件可以将数据依照所制定的格式逐一显示出来，这个预先定好的格式就是“模板”。Repeater 的模板同 GridView 控件的模板是类似的，但 Repeater 控件只能在源视图中进行编辑。

【例 4-38】 采用 SqlDataSource 和 Repeater 控件显示 student 表，详见 Ex4-38.aspx 页面(见资源包)。

【示例说明】

(1) 对 Repeater 控件的布局代码在源视图中进行。

(2) #Eval 功能是取得数据集内的指定内容，参数是字段名或属性名。

(3) 本例是使用 Table 来布局 Repeater 控件。Repeater 控件本身就是一个 Table，编程者需要对 HTML 的表格相关标记较为熟悉。

4.4.7 DataList 控件

DataList 控件同 Repeater 控件功能相同，但它除了显示数据的功能外，还提供数据更新和删除功能。DataList 控件也是依据模板的定制来展示数据的。与 Repeater 控件不同的是，DataList 控件的模板设置和 GridView 一样，可以通过可视化的编辑方式进行，其模板类型见表 4-16。

表 4-16 DataList 控件的模板

模板名称	说明
AlternatingItemTemplate	间隔行模板
EditItemTemlpate	编辑行模板
FooterTemplate	脚注模板
HeaderTemplate	标题模板
ItemTemplate	每个显示行模板
SelectedItemTemplate	选中行模板
SeparatorTemplate	替换项模板

默认情况下，DataList 的项目在表内以单个垂直列呈现，如果将其 RepeatLayout 属性设置成 Flow，将从列表的呈现形式中移除 HTML 表结构。

DataList 通过 RepeatDirection 属性可以以水平(Horizontal)或垂直(Vertical)显示项目。在显示时，可以通过设置 RepeatColumns 决定显示的列数。

【例 4-39】 采用 SqlDataSource 和 DataList 控件显示 student 表，详见 Ex4-39.aspx 页面(见资源包)。

【例 4-40】 采用 SqlDataSource 和 DataList 控件编辑 student 表，详见 Ex4-40.aspx 页面(见资源包)。

【示例说明】

（1）DataList 控件不像 GridView 控件通过设置自动编辑删除的属性就可以自动更新和删除。它需要手工在模板中编辑、删除、更新、取消等功能的按钮，并将其 CommandName 设置为 Edit（编辑）、Delete（删除）、Update（更新）、Cancel（取消），这样当单击相应的按钮时，DataList 会自动调用 EditCommand、DeleteCommand、UpdateCommand、CancelCommand 事件。

（2）EditCommand、DeleteCommand、UpdateCommand、CancelCommand 等事件的作用是分别执行相应编辑、删除、更新和取消功能，需要在后台这几个事件中编写代码完成操作。

（3）本例用的是 SqlDataSource 控件连接数据源，因此在更新（删除）时需要为 UpdateParameters（DeleteParameters）的参数提供参数，然后调用 SqlDataSource 的 Update()、Delete()方法。



本章小结

本章主要介绍了 ASP.NET 进行数据库编程的方法。首先介绍了数据库的基本概念和数据库的设计方法，然后讲述了 SQL 语句的使用方法以及 SQL Server 数据库的安装，接下来讲述了 ADO.NET 的对象模型和应用，最后讲述了 ASP.NET 的数据显示控件。

本章 SQL 语句是利用 ADO.NET 访问和数据库的语言基础，ADO.NET 对象模型是 ASP.NET 进行数据库编程的核心内容，ASP.NET 的数据显示控件则是展示和操作数据库信息的重要工具。限于篇幅，本章对各种对象控件的介绍还不够深入，有兴趣的读者请查阅相关资料进行更深一步的学习。



思考与练习

根据本章的介绍，利用 ASP.NET 编写一个数据库管理系统，实现数据的录入、修改、删除、查找等功能。

第 5 章

网 站 设 计

随着网络技术的发展，网站已经变得非常流行，几乎所有的企业和部门都有自己的网站，但是网站的建设却非常复杂，是一个综合性的系统工程。

在网站的建设过程中，要充分了解网站使用者的需求，在众多技术中选择适当的物理环境、开发技术和运行平台，要让美工、程序员和数据库管理员相互配合，还要考虑数据安全和技术升级，这就需要开发者在开发网站之前要进行非常详细的规划与设计。

本章将对网站设计进行详细介绍，旨在使读者对于网站建设系统工作有个全面的了解。

5.1 网站规划与设计

网络的发展给网站设计者提供了广阔的设计空间。相对传统的平面设计来说，网站设计具有更多的新特性和更多的表现手段，借助网络这一平台，将传统设计与计算机、互联网技术相结合，实现网站设计的创新应用与技术交流。

5.1.1 网站功能

网站根据使用的目的不同，一般分为四大类。

1. 形象宣传

以宣传网站所属单位为建站目的，用于提高单位的社会知名度，或是发布相关新闻或信息。例如，“中华人民共和国中央人民政府”网站就是中国政府对外的网上窗口，如图 5-1 所示。

2. 数据展示

以数据展示为建站目的网站一般适合于各种企业，其发布的信息主要是关于产品的价格、服务收费的标准、产品的规格、产品的数量等数据，如果用形象展示的方式就不合适。当然，这里提到的数据并不单指阿拉伯数字。



图 5-1 “中华人民共和国中央人民政府”网站

数据包括不同类型的数字、字符、计算公式等信息，还包括其他多媒体信息。图 5-2 所示的“中关村在线”网站就是一种数据展示网站。



图 5-2 “中关村在线”网站

3. 电子商务

电子商务网站与其他两类网站相比增加了在线交易功能，使得网站不但是相关单位的窗口，而且可以作为交易平台使用。图 5-3 所示的“京东”网站就是大型电子商务网站。



图 5-3 “京东”网站

4. 论坛

论坛类网站可以看作是一个社交平台，通过论坛可以对一些相关的信息与其他网络用户进行探讨。图 5-4 所示的就是“天涯社区”论坛网站。



图 5-4 “天涯社区”论坛网站

网站功能取决于需求者的要求，根据不同的诉求，选择、设计并建设适合的网站。

5.1.2 网站制作流程

网站制作的流程大致可以分为下列4个阶段。

1. 收集资料、规划网站架构

搜集数据与规划网站架构是网站制作的首要步骤，在这个阶段，除了要弄清网站所要传递的内容外，最重要的是必须确立网站的主题与目标群，然后将网站内容规划成分层式架构，也就是规划出组成网站的网页，并根据主题与目标群决定网页的呈现方式。

确立网站的主题与目标群是非常重要的，下列几个问题值得深思。

(1) 网站的搭建是为了销售产品或服务、塑造宣传企业形象还是方便业务联系或个人兴趣分享？如果网站本身具有商业目的，那么是否还需要进一步了解相关行业背景，包括产品类型、企业文化、品牌理念、竞争对手等。

(2) 网站的搭建与经营需要投入多少时间与资源？如何营销网站？有哪些渠道及相关的费用？

(3) 网站的获利模式是什么？如销售产品或服务、广告收益或其他。

(4) 网站将提供哪些资源或服务给哪些对象？关于这些对象，他们有哪些共同的特征或需求？比如，入口网站或购物网站的使用者比较广泛，所以其首页通常涵盖了琳琅满目的题材，而彩妆网站的使用者可能锁定为时尚女性，所以其首页往往呈现出艳丽的视觉效果，以便紧紧抓住浏览者的目光。

2. 网页制作与测试

在这个阶段中，需要着手制作阶段一所规划的网页，常见的网页编辑软件分为两种：①纯文本编辑软件，如记事本、WordPad等；②所见即所得的网页编辑软件，如Dreamweaver，而且必要时还要使用Photoshop、Flash等软件来设计网页背景、标题、按钮、动画等。网页制作完毕后，还要在计算机上测试页面上各个元素能否正确运行。

对于想学习HTML的人，使用纯文本编辑软件来编辑网页是较佳的选择，因为它可以让使用者专注于HTML的语法，不像所见即所得的网页编辑软件会产生一些多余或特有的HTML标签，造成初学者的困扰；相反，对于不想学习HTML只想快速编辑网页的人来说，所见即所得的网页编辑软件则是较佳的选择，因为它隔绝了使用者与HTML的语法，使用者即使不具备程序设计知识，也一样能设计出图文并茂的网页。

3. 将网站上传到Web服务器

辛苦制作的网站最后要上传到Internet让大家欣赏，此时需要先替网站在Internet上找个家，也就是申请网页空间，常见的方式如下。

(1) 租用专线。如果你的预算充足，可以向中国电信、中国移动、中国联通、长城宽带等ISP租用专线，让你的计算机24小时上网并架设成Web服务器，其他人就能通过Internet浏览你的网站了。

(2) 租用网页空间或虚拟主机。ISP通常会提供网页空间或虚拟主机出租业务，这种业务的价格较低，适合预算少的人。有关网页空间或虚拟主机的出租价格、申请程序、上传方

式、网页空间大小、传输速率等事项，可到 ISP 的网站上查询，或者到百度等搜索网站搜索。

4. 网站更新与维护

网站开发人员的责任不仅仅是将网站上传到 Internet 就结束了，还需要负起维护与更新的责任，才能使网站有更多的访问量，产生社会或经济效益。这就需要定期更新网页内容，然后通过网页空间提供者所提供的界面或 FTP 软件，将更新后的网页上传到 Internet，同时还要不时上网检查一下网站的运行是否正常，保证网站数据的安全。

5.1.3 网站需求分析

网站开发人员要开发出适合的网站系统，就必须要了解和掌握网站的开发目的、网站的功能和拟投入的开发费用等信息，因此要进行必要的需求分析。

需求分析的因素主要包括以下几个。

1. 网站应用定位

网站应用定位就是要了解待开发网站的使用目的，如商业性质的网站就要考虑其市场环境。以电子商务网站为例，其最终目的是通过网站宣传企业的产品或服务，销售企业的产品或服务，提升企业的知名度，获取利润。网站的应用定位主要包括网站使用者的信息，如年龄、学历、收入甚至个人偏好等，还要考虑当时的市场环境与国家政策。

2. 人员

不管何种网站，人员的配置和合作开发都是网站开发过程中重要的环节。

3. 资金

资金是选择建设网站技术水平的重要支柱。一些软件需要企业大量的投入，保证运行的安全和稳定。在进行建设网站的方案论证时要把资金要素放在重要位置进行考虑，毕竟一个网站从建设、测试到运行，是一个持续的过程，不能中断中间的资金投入。

4. 技术

网站开发时所涉及的技术众多，在各种相关技术中，如何选择以及如何综合使用这些技术，是网站开发成功与否的关键因素。关键技术主要有下几个。

1）网页制作技术

由于网页是用户访问网站首先接触到的内容，所以有人说，网页是网站的核心内容。也有人说，网页就是浏览器窗口出现的文件，当用户使用 URL 时，该文件被调用，并且出现在窗口中。可见，对于用户来说网页十分重要。

2）开发语言

开发语言是网站尤其是动态网站的核心，除了包括前面章节介绍的 ASP.NET 以外，还可以使用 PHP、ASP、JSP 甚至 CGI（通用网关接口）技术。

3）数据库技术

数据库技术是实现电子商务的重要技术支撑。它主要包括硬件平台、软件平台、数据库管理人员和数据库用户 4 个方面。由于企业要创建自己的电子商务网站，因此一方面需要构筑自己的数据库系统，包括企业开发数据库所需的计算机设备、开发所需的操作系

统、数据库开发工具、开发与管理人员、与前台链接的数据库技术等。

另一方面，还要具备登录到 Internet 后共享其他资源时对数据库技术的各种需要，如共享资源所需要的硬件设备、软件平台、掌握大型数据库使用方法并熟悉网络操作系统的人员等。

4）安全技术

创建网站需要很高的安全技术作为保障，安全问题也是阻碍网站尤其是商业类网站（如电子商务网站）发展的重要因素。由于 Internet 具有集成、松散和开放的特点，网站容易受到黑客的攻击，他们窃取商业信息，甚至破坏系统。在这样一个环境中，网站必须考虑各种保证安全的措施，如可以采取的措施包括防火墙技术、密码技术、数字签名技术、数字时间戳、数字凭证认证等。

5.1.4 网站总体设计

网站总体设计是对网站中所有网页整体布局和色彩等内容进行的整体规划和设计，使整个网站的风格更加统一，色彩搭配更加合理，界面更友好。其主要包括网站整体布局、页面布局的具体方法和网页的色彩搭配的设计，通过网页风格的设计使网页布局趋于合理，色彩搭配更加赏心悦目。

在现代网页设计过程中，网页布局设计变得越来越重要。网页的浏览者不愿意再看到只注重内容的站点。虽然内容仍然是最重要，但只有当网页布局和网页的内容有机地结合在一起时，这种网页或者说站点才是最受浏览者欢迎的。如果过于偏重内容而忽略了网页的整体布局，或者网页的布局异常华丽而没有实质性的内容，都无法留住越来越“挑剔”的访问者。

网页是网站的重要组成部分，其通过友好、美观的界面与网站使用者进行交互，是影响网站访问效果的最主要因素。

网站要容易使用，并遵循一些设计规则，因为用户在访问一个网站时，难以记住任何特殊的交互诀窍，于是会把更多的时间花在更容易使用的网站上。因此在设计网站的页面时，遵循这些原则显得尤为重要。

主页经常是第一个（也有可能是最后一个）吸引并留住访问者的机会，在这点上主页和报纸的头版非常相像。所有的主页都应该像报纸的头版那样着重对待，并由编辑来决定主页上登载的重要内容和保证网站风格的一致性、连续性。

当然，这些只是指导性原则，而不是放之四海而皆准的公理。凡事皆有例外，虽然按照这些原则能大大提高主页的作用，但网页的设计人员还需要和具体用户进行广泛的交流，学习相关领域的知识，了解客户的需求，并进行实际用户测试，把不断反馈的信息融入开发周期中。只有在不断的反馈修改的过程中才能制作出会用户满意的网页。

5.2 页面美化技术

HTML 作为网页显示的基础元素，只针对内容进行相应的处理，但无法提供更友好、美观、方便的页面显示，虽然图片和 Flash 技术的应用可以弥补这些缺憾，但是过多地使

用，将降低访问速度。

CSS 的引入大大简化了页面美化技术的复杂度，提高了网站的访问速度，并增加了许多可扩展功能。

5.2.1 CSS 的使用

CSS(Cascading Style Sheet，层叠样式表)简称样式表。样式就是对网页中的元素(文字、段落、图像、列表等)属性显示形式的描述。例如，文字的大小、字体、背景及图像的颜色、大小等都是样式，这些样式被一项项写在样式表中。层叠就是指当 HTML 文件引用多个样式表文件时，如果这些文件之间所定义的样式发生了冲突，将依据元素的包含层次来处理样式对内容的控制。CSS 就是一系列控制网页元素外观的样式规则。

CSS 对网站的开发者和浏览者有着重要的贡献。主要体现在以下几个方面。

(1) 内容和样式的分离，使网页设计趋于明了、简洁。

(2) 弥补 HTML 对标记属性控制的不足，如背景图像重复的控制和标题大小的控制等。在 HTML 中可控制的标题仅有 7 级，即 h1～h7，而利用 CSS 可以任意设置标题大小。

(3) 精确控制网页布局，如行间距、字间距、段落缩进和图片定位等属性。

(4) 提高网页效率，增强易用性和可扩展性。多个网页同时应用一个 CSS 样式，既减少了代码的下载，又提高了浏览器的浏览速度和网页的更新速度，并能提高网页的编辑维护效率。同时，在一个网页上可以通过调用不同的样式表来切换不同的网页外观。

(5) 增强网页特效，提高用户体验。CSS 还有许多特殊功能，如鼠标指针属性控制鼠标的形状，滤镜属性控制图片的特效等。

CSS 的这些作用使得它成为网页技术发展的必然。从 20 世纪 90 年代初 HTML 被发明开始，样式表就以各种形式出现了，不同的浏览器结合了它们各自的样式语言，呈献给读者符合其样式规则的网页内容，为了减弱这种情况对读者造成的困扰，W3C 组织联合 CSS 的创始者于 1996 年 12 月出版 CSS 的第一版。

1997 年年初，W3C 内组织了专门管 CSS 的工作组，开始讨论第一版中没有涉及的问题，并于 1998 年 5 月出版 CSS 第二版。而后，W3C 又致力于 CSS 第三版的研究。CSS3 是由一系列标准的模块构成，很多模块现今仍处于开发阶段。


【小提示】

为了检查网站的标准性，W3C 提供了网站的检查功能，验证地址 http://validator.w3.org/，主要用于检查 XHTML 代码的书写是否规范和检查 CSS 的属性是否都在 CSS2 范围内。

CSS 和 HTML 一样，也是用文本书写的一些规则，其创建和编辑也常用 Dreamweaver 等可视化的网页编辑软件，或者使用记事本直接书写。

样式表中的样式是针对网页元素书写的，只有将样式应用在相应的网页元素上，并通过浏览器显示出来，才能让用户看到效果。有 3 种方法可以在站点网页上使用 CSS 样式。

1. 内联样式

将样式直接写在各个网页元素的标签中，这种样式只对样式所在的元素起作用。

【基本语法】

```
style = "属性: 属性值; "
<head>
⋮
</head>
<body>
⋮
<HTML 标记 style = "样式属性:取值;样式属性:取值;...">
⋮
</body>
```

【语法说明】

（1）HTML 标记就是页面中标记 HTML 元素的标记，如 body、p 等。

（2）style 参数后面引号中的内容就相当于样式表大括号里的内容。需要指出的是，style 参数可以应用于 HTML 文件中的 body 标记，以及除了 basefont、param 和 script 之外的任意元素。

【例 5-1】 在浏览器中显示的效果如图 5-5 所示。

```
<html>
<head>
<title>HTML 中使用 CSS 的方法 -- 内联样式</title>
</head>
<body>
<h1 style = "color:green;font - size:35px;font - family:黑体"> HTML 中使用 CSS 的方法</h1>
</body>
</html>
```

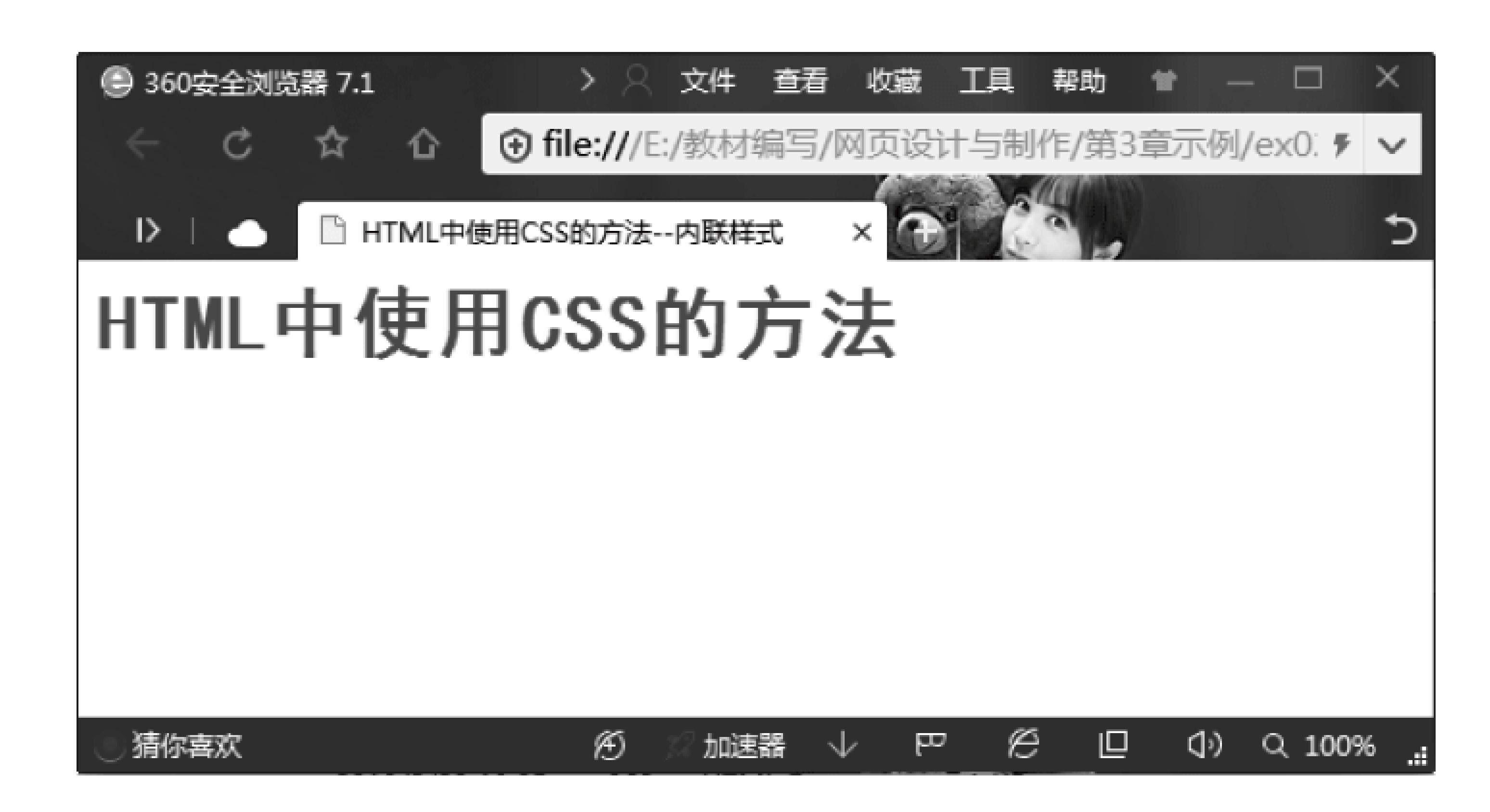


图 5-5 内联样式显示效果

【小提示】

为节省篇幅，后面的示例中将不再给出显示效果的截图，读者可以自行运行后查看其运行效果。

2. 内部样式

在网页<head>标签中创建嵌入的样式表，这种样式只针对样式表所在网页上的元素起作用。该方法将CSS样式用<style>标签标记，嵌套在<head>标签中。

【基本语法】

```
<head>
<style type="text/css">
<!--
选择符{样式属性:取值;样式属性:取值;...}
选择符{样式属性:取值;样式属性:取值;...}
⋮
-->

</head>
```

【语法说明】

(1) <style>标记用来说明所要定义的样式。

(2) type="text/css"说明这是一段CSS样式表代码。

(3) <!--与-->标记的加入是为了防止一些不支持CSS的浏览器，将<style>与</style>之间的CSS代码当成普通的字符串显示在网页中。

(4) 选择符也就是样式的名称，这里的选择符可以选用HTML标记的所有名称。

内部样式表方法就是将所有的样式表信息都列于HTML文件的头部，因此这些样式可以在整个HTML文件中调用。如果想对网页一次性加入样式表，即可选用该方法。

【例5-2】 请参看资源包中的例5-2。

3. 外部样式

将样式写在单独的一个文件中，保存为.css文件，通过链接或者导入的方式将外部样式表应用到网页上，这种方式可以使一个样式表作用在不同的网页上。

该方法将CSS样式写在单独的样式文件中，在网页<head>标签中可以通过<link>标签调用或者@import导入该样式文件。

1) 链入外部样式表

当要在站点上所有或部分网页上一致地应用相同样式时，可使用外部样式表。在一个或多个外部样式表中定义样式，并将它们链接到所有网页，便能确保所有网页外观的一致性。如果人们决定更改样式，只需在外部样式表中修改一次，而该更改会反映到所有与该样式表相链接的网页上。

【基本语法】

```
<head>
⋮
<link rel="stylesheet " type="text/css" href="样式表文件的地址 ">
</head>
⋮
```

【语法说明】

(1) rel="stylesheet"是指在 HTML 文件中使用的是外部样式表。

(2) type="text/css"指明该文件的类型是样式表文件。

(3) href 中的样式表文件地址,可以为绝对地址或相对地址。

(4) 外部样式表文件中不能含有任何 HTML 标签,如<head>或<style>等。

(5) CSS 文件要和 HTML 文件一起发布到服务器上,这样在用浏览器打开网页时,浏览器会按照该 HTML 网页所链接的外部样式表来显示其风格。

【小提示】

一个外部样式表文件可以应用于多个 HTML 文件。当改变这个样式表文件时,所有网页的样式都随之改变。因此常用在制作大量相同样式的网页中,因为使用这种方法不仅能减少重复工作量,而且方便以后的修改和编辑,有利于站点的维护。同时在浏览网页时一次性将样式表文件下载,减少了代码的重复下载。

外部样式表优点如下。

- 独立于 HTML 文件,便于修改。
- 多个文件可以引用同一个样式表文件。
- 样式表文件只需下载一次就可以在链接了该文件的页面内使用。

浏览器先显示 HTML 内容,然后再根据样式表文件进行渲染,从而使访问者可以更快地看到内容。

2) 嵌入外部样式表

当人们只是要定义当前网页的样式,可使用嵌入的样式表。嵌入的样式表是一种级联样式表,“嵌”在网页的<HEAD>标记符内。嵌入的样式表中的样式只能在同一网页上使用。

【基本语法】

```
<head>
<styletype="text/css">
@import  url(外部样式表文件地址);
⋮
</stytle>
⋮
</head>
```

【语法说明】

(1) Import 语句后面的“;”是不可省略的。

(2) 外部样式表文件的文件扩展名必须为.css。

(3) 样式表地址可以是绝对地址,也可以是相对地址。

【例 5-3】 请参看资源包中的例 5-3。



【小提示】

在使用中,某些浏览器可能会不支持导入外部样式表的@import 声明。所以此方法不经常用到。如果网页链接到外部样式表,为网页所创建的内嵌的或嵌入式样式将扩充或覆盖外部样式表中的指定属性。

5.2.2 CSS 基础语法

外部样式表中的内容使得页面的外观发生明显的变化,也就是 CSS 的描述部分或者叫定义部分。通常情况下,CSS 的描述部分是由 CSS 的选择器来承担的。

CSS2 选择器的基本语法如下。



【基本语法】

```
selector {property: value; property: value ...property: value }
```



【语法说明】

(1) 语法中 selector 代表选择符,property 代表属性,value 代表属性值。每一组属性和值称为一个声明。

(2) 属性和值要用冒号隔开;每一个声明要用分号结束。

(3) 选择符包括多种形式,所有的 HTML 标记都可以作为选择符,如 body、p、table 等都是选择符。具体将在下面讲解。

(4) 如果属性值由多个单词组成,并且单词间有空格,那么必须给值加上引号。

CSS 选择符就是 CSS 样式的名字。选择符的命名规则可以使用英文字母的大写与小写、数字、连字号、下划线、冒号、句号,CSS 选择符只能以字母开头,选择符在 CSS 中可分成多种,主要包括类型选择符、类选择符、ID 选择符、派生选择符、伪类等。

1. 类型选择符

类型选择符是现有的 HTML 标签(或称标记),也被称为标签选择符。用 CSS 控制这些标签,会改变标签的默认样式,如例 5-1 中的关于 h1 的样式选择符就是类型选择符。

【例 5-4】 请参看资源包中的例 5-4。

2. 类选择符

当需要对一个网页中的部分相同标签进行一样的样式设置时,则需要给这些标签一个新的别名,并对这些具有新别名的标签进行样式设置,以满足设计需求,这种样式的选

择符被称为类选择符。用类选择符可以把相同的元素分类定义成不同的样式。

进行类选择符类型定义的时候分为两步。

(1) 为标签定义新别名,即在标签后面书写"class=新别名"。类型选择符名称不区分大小写,但是类选择符与ID选择符名称区分大小写。类选择符与ID选择符名称应该由大写字母开始,随后使用任意字母、数字、连接线和下划线。常用的网页元素名称大多是根据网页元素的位置和作用命名的,常用命名方法如 news_title、NewsTitle 等。

(2) 定义新的类选择符样式。定义方法是将选择符名称写为".新别名"。

【例 5-5】 请参看资源包中的例 5-5。

3. ID 选择符

在 HTML 文档中,需要唯一标识一个元素时,就会赋予它一个 ID 标识,以便在对整个文档进行处理时能够很快地找到这个元素。而 ID 选择符就是用来对这个单一元素定义单独的样式。其定义方法与类选择符大同小异,命名时需要把 class 改为 ID,样式定义时只需要把"."改为"#"。

ID 标识只能在网页中出现一次,而定义的 ID 样式也只能在同一网页中使用一次。

【例 5-6】 请参看资源包中的例 5-6。

在应用选择符的过程中,可能会遇到同一个元素由不同选择符定义的情况,这时就要考虑到选择符的优先级。通常使用的选择符包括 ID 选择符、类选择符、派生选择符和类型选择符等。因为 ID 选择符是最后被加到元素上的,所以优先级最高,其次是类选择符。! important 语法主要用来提升样式规则的应用优先级。

4. 派生选择符

当仅仅想对某一元素中的子元素进行样式指定时,派生选择符就被派上了用场,派生选择符指选择符中前一个元素包含后一个元素,元素之间使用空格作为分隔符。

【例 5-7】 请参看资源包中的例 5-7。

5. 伪类

伪类不属于选择符,它是让页面呈现丰富表现力的特殊属性。之所以称为"伪",是因为它指定的对象在文档中并不存在,它们指定的是元素的某种状态。

应用最为广泛的伪类是链接的 4 个状态。

(1) 未访问链接状态(a:link)。

(2) 已访问链接状态(a:visited)。

(3) 鼠标指针悬停在链接上的状态(a:hover)。

(4) 被激活(在鼠标单击与释放之间发生的事件)的链接状态(a:active)。

【例 5-8】 请参看资源包中的例 5-8。

5.2.3 CSS 常用单位

CSS 常用单位包括长度单位和颜色单位。

1. 长度单位

长度单位是 Web 页设计中最常用的一个单位。在 CSS 中,长度是一种度量尺寸,用

于宽度、高度、字号、字和字母间距、文本的缩排、行高、页边距、贴边、边框线宽以及许多的其他属性。

长度单位一般是一个由两个字母组成的单位缩写，如 cm、pt、em 等。

1）绝对长度单位

网页定义上常常使用的绝对长度单位有厘米（cm）、毫米（mm）、英寸（in）、点（pt）、派卡（pc）等，如表 5-1 所示。绝对长度值的使用范围比较有限，只有在完全知道外部输出设备的具体情况下才使用绝对长度值。也就是说，绝对长度值最好用于打印机输出设备，而在仅仅作为屏幕显示时，使用绝对长度值意义不大，应该尽量使用相对长度值。

表 5-1　绝对长度单位

单位	描述	单位	描述
in	英寸	pt	磅（1pt=1/72in）
cm	厘米	pc	12 点活字（1pc=12 点）
mm	毫米		

2）相对长度值

每一个浏览器都有其自己默认的通用尺寸标准，这个标准可以由系统决定，也可以由用户按照自己的爱好进行设置。因此，这个默认值尺寸往往是用户觉得最适合的尺寸。于是使用相对长度值，就可以使需要定义尺寸的元素按照默认大小为标准，相应地按比例缩放。这样就不可能产生难以辨认的情况。

CSS 支持下列 3 种长度的相对单位：em（当前字体中字母 M 的宽度）、ex（当前字体中字母 X 的高度）、px（一个像素的大小），如表 5-2 所示。

表 5-2　长度的相对单位

单位	描　　述
em	1em 等于当前的字体尺寸 2em 等于当前字体尺寸的 2 倍 例如，如果某元素以 12pt 显示，那么 2em 是 24pt 在 CSS 中，em 是非常有用的单位，因为它可以自动适应用户所使用的字体
ex	一个 ex 是一个字体的 x-height（x-height 通常是字体尺寸的一半）
px	像素（计算机屏幕上的一个点）

使用 em 和 ex 的目的就是为所给的字体设置合适的宽度，而没有必要知道字体有多大，在显示时，可通过比较当前字体中的 M 和 X 来确定。字体越大，所对应的 em 和 ex 也就越大。

以像素为单位的长度是相对于显示器上像素（或许为方形）的高度和宽度。影像的宽度和高度经常是以像素给出的。像素测量法通常不是个好方法。首先，像素的大小依分辨率变化较大，大多数用户都会将显示器设置成尽可能高的分辨率，从而导致像素太小而无法阅读。

3）百分比

百分比总是相对于另一个值来说的，那个值可以是长度单位或是其他的。每一个可以使用百分比值单位指定的属性同时也自定义了这个百分比值的参照值。大多数情况下，这个参照值是此元素本身的字体大小。要使用百分比，首先应该在所选择的选项后面的文本框中写符号部分，这个符号可以是“＋”（正号），表示正长度值，也可以是“－”（负号），表示负长度值。如果不写符号，那么默认值是“＋”。

在符号后紧接着的是一个数值，符号后面可以输入任意值，但是由于在某些情况下，浏览器不能处理带小数的百分数，因此最好不用带小数的百分比。然后再在该选项的长度单位下拉列表框中选择“％”。

2. 颜色单位

定义颜色值最简单、最直接的方法是使用百分比值。在这种情况下，红、绿、蓝颜色值的等级用百分比数来确定。格式是 rgb(R％,G％,B％)。使用百分比值来指定颜色还有一个好处，是它能够声明一组真正的数字，不论它的值是多少。

指定颜色的另一种方法是使用范围在 0～255 之间的整数来设置。格式是 rgb(128，128，128)。

定义颜色的第三种方法是使用十六进制数组定义颜色。这种定义方法对于经常进行程序设计的人来说比较熟悉。定义颜色时使用 3 个前后按顺序排列的十六进制数组表示，如"＃FC0EA8"。这种定义的方式就是形如＃RRGGBB 的格式，即在红、绿、蓝的位置上添上需要的十六进制值。

定义颜色最后一种方法也是最简单的方法是指定颜色的名称，也称颜色关键字。在 CSS 的颜色定义中，总共有 147 种颜色关键字。详细内容可查阅 CSS 手册，如表 5-3 所示。

表 5-3　颜色单位

单　　位	描　　述
(颜色名)	颜色名称(如 red)
rgb(x,x,x)	RGB 值[如 rgb(255,0,0)]
rgb(x％,x％, x％)	RGB 百分比值[如 rgb(100％,0％,0％)]
＃rrggbb	十六进制数(如＃ff0000)

5.2.4　CSS 字体属性

在 CSS2 中提供了很多关于字体设置的样式，使用它们可以很容易地设置字体的类型、大小、颜色等效果。

1. 字体属性

1）font-family

计算机的系统里预装了很多字体，使用 font-family 属性可以选择开发者想要使用的字体。

【基本语法】

```
font - family:字体 1,字体 2...;
```

【语法说明】

其中字体 1 是优先选择的字体，如果该字体在用户的系统中不存在，那浏览器就调用字体 2 指定的字体，以此类推。

2）font-size

该属性可以设置字体的绝对大小或者相对大小。

【基本语法】

```
font - size:绝对大小或者相对大小;
```

【语法说明】

绝对大小为用户指定文字的绝对大小，通常使用 pt 或者 in 作单位，但是会产生不同浏览器文字大小不一样的情况，造成视觉困扰；相对大小一般采用百分比的形式呈现，可以使页面的文字大小不管在何种浏览器下都能体现不同等级。

3）font-weight

该属性用来设置字体的加粗情况。

【基本语法】

```
font - weight:normal|bold|bolder|lighter|100 - 900;
```

【语法说明】

值中的前 4 项是系统给出的值。normal 是正常值，不加粗；bold 为粗体，字体粗细约为 700；bolder 比粗体更粗些，约为 900；lighter 比默认的字体细些。100～900 数值越小表示字体越细，数值越大字体越粗。

4）font-style

该属性用来设置字体的倾斜效果。

【基本语法】

```
font - style:normal|italic|oblique;
```

【语法说明】

其中，normal 为正常字体，不倾斜，是默认值；italic 为倾斜；oblique 为偏斜体。

5）font-variant

该属性用来将中文字体转换为较小的中文字体，将英文字体转换为大写且字体较小的英文字体。

【基本语法】

```
font-variant:normal|small-caps;
```

【语法说明】

其中，normal 为默认值，是正常的字体；small-caps 是将小写英文字体转为大写英文字体。

6）font

上述各项是对字体的某一项属性进行设置，在 CSS 中提供了 font 属性，将大部分关于字体的属性集中在这一个属性上，用来对字体进行综合设置。

【基本语法】

```
font:font-family  font-style  font-variant  font-weight  font-size;
```

【例 5-9】 请参看资源包中的例 5-9。

2. 文字样式属性

1）color

该属性用来设置文字的颜色。

【基本语法】

```
color:value;
```

【语法说明】

上面介绍了颜色的 4 种单位：颜色名称、RGB 值、RGB 百分比值、十六进制数，在 color 属性设置时，value 值可以是这 4 种单位中的任何一种。

2）line-height

该属性用来设置行高，控制行间的距离。

【基本语法】

```
line-height:normal|比例|长度单位|百分比;
```

【语法说明】

其中，normal 是默认值；比例是指相对于元素 font-size 设定大小的倍数；长度单位可利用绝对以及相对的值来设置高度；百分比是相对于元素 font-size 的百分比。比例、长度单位、百分比的值可以为正值，也可以为负值。正值表示距离拉大，负值表示距离减少。

3）word-spacing

该属性用来设置英文单词之间的距离。

【基本语法】

```
word - spacing: normal|比例|长度单位|百分比;
```

该属性中的各值与 line-height 属性的值是一样的。

4）letter-spacing

该属性用来设置字母之间的距离。

【基本语法】

```
letter - spacing: normal|比例|长度单位|百分比;
```

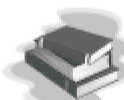
该属性中的各值与 line-height 属性的值也是一样的。

5）text-shadow

该属性用来设置文本阴影的颜色和位置。

【基本语法】

```
text - shadow:颜色值  长度值 1  长度值 2  长度值 3;
```

【语法说明】

其中，颜色值用来设置阴影的颜色；长度值 1 可以是绝对值也可以是相对值，用来设置阴影在 *X* 轴方向相对于文本本身位置的偏移距离；长度值 2 用来设置 *Y* 轴的偏移距离；长度值 3 用来设置阴影的模糊半径的值，值越大阴影就越分散。

6）text-decoration

该属性用来对文本进行修饰，如添加下划线、顶线、删除线、文字闪烁等效果。

【基本语法】

```
text - decoration:underline|overline|line - through|blink|none;
```

【语法说明】

其中，none 为默认值，不加任何效果；underline 为文字添加下划线；overline 为文字添加上顶线；line-through 为文字中间添加删除线；blink 为文字添加闪烁效果。

【例 5-10】 请参看资源包中的例 5-10。

5.2.5 CSS 段落属性

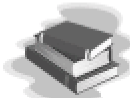
在 CSS2 中提供了很多关于段落设置的样式，使用它们可以很容易地设置段落的排版格式等效果。

1. text-align

该属性用来控制文字段落的水平对齐方式。

【基本语法】

```
text - align:left|right|center|justify;
```

【语法说明】

其中，left 为左对齐；right 为右对齐；center 为居中对齐；justify 为两端对齐。

2．text-transform

该属性用来转换英文字母的大小写形式。

【基本语法】

```
text - transform:capitalize|uppercase|lowercase|none;
```

【语法说明】

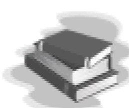
其中，capitalize 将每个英文单词的首字母大写；uppercase 将每个英文字母转换为大写；lowercase 将每个英文字母转换为小写；none 默认值，不改变大小写。

3．text-indent

该属性用来设置段落的首行缩进。

【基本语法】

```
text - indent:value;
```

【语法说明】

其中，value 值可以是长度单位，绝对单位或者相对单位都可以，相对的是文本元素 font-size 大小；value 值也可以是百分比单位。value 值可有正负，正值表示向右缩进，负值表示向左突出。

【例 5-11】 请参看资源包中的例 5-11。

4．vertical-align

该属性用来设置段落文本的垂直对齐方式。

【基本语法】

```
vertical - align: baseline |sub | super |top |text - top |middle |bottom |text - bottom |length
```

【语法说明】

其中，baseline 表示将支持 valign 特性的对象的内容与基线对齐；sub 表示垂直对齐文本的下标；super 表示垂直对齐文本的上标；top 表示将支持 valign 特性的对象的内容与对象顶端对齐；text-top 表示将支持 valign 特性的对象的文本与对象顶端对齐；middle 表示将支持 valign 特性的对象的内容与对象中部对齐；bottom 表示将支持 valign

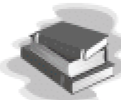
特性的对象的内容与对象底端对齐；text-bottom 表示将支持 valign 特性的对象的文本与对象底端对齐；length 表示 CSS2 由浮点数字和单位标识符组成的长度值|百分数，可为负数，定义由基线算起的偏移量。基线对于数值来说为 0，对于百分数来说就是 0%。目前 IE 5 尚不支持。

5. direction

该属性用于设置文本块的方向或书写方向。

【基本语法】

```
direction : ltr | rtl | inherit ;
```

【语法说明】

其中，ltr 表示文本从左到右书写，是默认值；rtl 表示文本从右到左书写；inherit 表示文本的值不可继承。

【例 5-12】 请参看资源包中的例 5-12。

6. :first-letter

这是一个伪类，用来设置段落的首字的样式。

【基本语法】

```
选择符:first-letter{属性值: 值;...}
```

该样式中常用的属性有 font、color、background、word-spacing、letter-spacing、text-decoration、vertical-align、text-transform、line-height、clear。

7. :first-line

这也是一个伪类，用来设置段落的首行文字的样式。

【基本语法】

```
选择符:first-line{属性值: 值;...}
```

样式中常用的属性与:first-letter 相同。

【例 5-13】 请参看资源包中的例 5-13。

5.2.6　CSS 定位

元素定位是 CSS 布局的基础，只有将各个网页元素放置到合适的位置，才能合理构建出网页的布局。在 CSS 中，实现元素定位的方法主要有两种：利用定位属性定位和利用浮动属性定位。

1. 定位属性

利用定位属性(position)实现元素定位主要包括 3 种方式：绝对定位、相对定位和固定定位。必须注意的是，在利用定位属性进行定位时，还需结合边偏移属性和 z-index 属

性来实现。其中定位属性用来确定采用哪种定位方式，边偏移属性用来确定元素的位置，而 z-index 属性用来确定元素的层叠顺序。

1）定位属性

【基本语法】

```
position :   static| absolute | relative | fixed;
```

【语法说明】

（1）static：静态定位。默认取值，即页面中的普通元素默认均为静态定位。

（2）absolute：绝对定位。

（3）relative：相对定位。

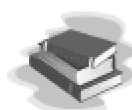
（4）fixed：固定定位。

2）边偏移属性

边偏移属性主要包括 4 个方向的分量，分别是 top（上）、right（右）、bottom（下）和 left（左）。

【基本语法】

```
top|right|left|botton:值
```

【语法说明】

（1）top：定义元素相对于父元素上边线的距离。

（2）right：定义元素相对于父元素右边线的距离。

（3）bottom：定义元素相对于父元素下边线的距离。

（4）left：定义元素相对于父元素左边线的距离。

一般在水平方向和垂直方向各选取一个分量即可准确控制元素的位置。例如，在元素宽、高都确定的情况下，设置边偏移中的 top、left 两个属性即可确定各个元素在页面中的确切位置。

3）z-index 属性。z-index 属性用以控制元素重叠时的顺序关系。取值可以是正数，也可以是负数，默认取值为 0。取值越大，则在层叠关系中越处于上层；反之，则处于下层。

2. 绝对定位

绝对定位属性的写法为 position：absolute。它是以父元素的坐标原点为参照，通过边偏移属性来控制元素位置的定位方式。一般在页面布局中父元素为 body，因此参照的坐标原点在浏览器的左上角。

页面中的各个元素使用绝对定位之后，即脱离了普通文本流，因此可以将其精确地定位到页面中的任意位置。Dreamweaver CS5 中的 AP 元素即是采用绝对定位之后的 div 元素。

【例 5-14】 请参看资源包中的例 5-14。

3. 相对定位

相对定位属性的写法为 position：relative。相对定位的参照原点是元素本身在默认

情况时(即 position 值为 static 时)的位置。要注意的是,元素相对定位后位置发生了偏移,但是元素原本占有的位置并未消失。

【例 5-15】 请参看资源包中的例 5-15。

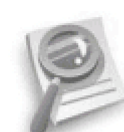
4. 固定定位

固定定位属性的写法为 position: fixed。采用这种定位可使元素固定显示于屏幕中的某个位置。当拖动浏览器的滚动条阅读其他内容时,固定定位的元素位置不变。对于这种定位方式,某些浏览器不支持(如 IE 6 及其以下版本)。

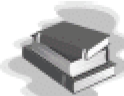
5. 浮动属性

浮动(float)是最常用的一种元素定位方式。与利用定位属性的方式相比,浮动更方便、直观。它能改变元素的默认显示方式,可使原本换行显示的块元素转为同行显示,由此来实现对块元素的定位。DIV+CSS 的页面布局方式也主要是基于此来实现。

浮动属性用来设置元素是否浮动以及其具体的浮动方式,它的语法如下。

**【基本语法】**

```
float:none|Left|right;
```

**【语法说明】**

(1) none:不浮动。页面中的元素默认都不浮动。

(2) left:元素向左浮动。

(3) right:元素向右浮动。

页面中的各个元素若都设置为左浮动,则按定义顺序依次从页面左侧开始排列;若设置为右浮动,则依次从页面右侧开始排列。如果当前浏览器的宽度不足以显示所有的块,则后续的块会自动换行向左侧或右侧排列。

例 5-16 给出了两个元素,一个左浮动,另一个右浮动的例子。

【例 5-16】 请参看资源包中的例 5-16。

5.3 页面动态技术

只使用 HTML 和 CSS 技术的网页,页面在浏览时只能静态地显示给访问者,如果需要让页面动起来,或是让页面可以与访问者交互,就需要前台脚本技术。

**【小提示】**

本节介绍的技术虽然也称为动态技术,但主要针对的是页面的动态显示效果,而本书所说的动态,则是指与服务器尤其是数据库的动态交互技术。随着新技术的发展,前台动态技术也可以完成与后台数据库服务器的数据交互,如 Ajax 技术。

前台脚本技术主要包括两种实现,一个是基于 Java 语法规划的 JavaScript 脚本语言,另一个是基于 VB 语法规划的 VBScript。两种脚本都可以实现页面动态效果,本节以

JavaScript 为例加以讲解。

5.3.1 JavaScript 概述

JavaScript 是一种以网页为基础，基于对象和事件驱动，主要对网页进行修饰，并可以与服务器端程序进行通信的，源于 Java 语言的客户端脚本语言。最早由 Netscape 开发的 Live Script 发展而来，主要目的是解决服务器端语言，比如 Perl 或其他 CGI(通过用网接口)程序遗留的速度和频繁交互的问题，为客户端提供更流畅、更方便的使用效果。JavaScript 可以通过其对客户端网页页面进行修饰，如增加动态的菜单效果、增加窗口的显示效果、对网页中的各种对象进行控制等，都可以使用 JavaScript 完成，它是微软公司开发的 VBScript 的有力竞争对手，现在很大一部分的网页都是使用其进行开发与修饰的。

JavaScript 的正式名称是 ECMA Script，是由 ECMA 组织发展和维护。ECMA-262 是正式的 JavaScript 标准，这个标准基于应用于 Netscape 浏览器的 JavaScript 和应用于 Microsoft 浏览器的 JScript 而制订的。

Netscape 公司的 Navigator 2.0 的开发者之一 Brendan Erich 发明了这门语言，并从 1996 年开始其已经出现在所有的 Netscape 和 Microsoft 浏览器中，在 1997 年 7 月的 ECMA 会员大会上采纳了其首个版本，在 1998 年，其成为国际 ISO 标准(ISO/IEC 16262)。

JavaScript 是嵌入在 HTML 源代码中的，因此可以使用任何一款文本编辑工具(如记事本、Dreamweaver 或是 Ultra Edit 等)进行编写，然后对 JavaScript 提供支持的浏览器中运行就可以看到效果。

【例 5-17】 请参看资源包中的例 5-17。

JavaScript 的语句书写类似 Java 或 C 语言，语法规则也很相似，通常要在每行语句的结尾处加上一个分号，但根据 JavaScript 标准，分号是可选的，浏览器也可以把行末作为语句的结尾，但需要注意，如果在一行上写下多条语句里，每条语句的结尾处必须使用分号。

JavaScript 虽然是一个嵌入 HTML 源代码中的程序，但它不属于 HTML 中的内容，因此 JavaScript 的代码要放在 HTML 标记符<script>和</script>之间；否则浏览器不会对 JavaScript 代码进行解释和执行。



图 5-6 Dreamweaver 的代码编辑环境里 language 选项的可选值

<script>标记符中有多个参数，如果包含的是 JavaScript 代码，其 language 选项的值要设置为 JavaScript。此外，也可以设置为 JavaScript1.1、JavaScript1.2 或是 jscript 等值，其含义也都表示支持 JavaScript 脚本语言。在 Dreamweaver 中的代码编辑环境里，可以看到 language 选项的可选值，其中以 j 开头的就都是支持 JavaScript 脚本的选项，如图 5-6 所示。

在一个网页中，并没有规定只能使用一次

<script>标记符,因此有些网页中既在<head>和</head>之间包含了 JavaScript 代码,也在<body>和</body>之间包含了 JavaScript 代码。

JavaScript 代码除了可以直接放在<script>和</script>标记符之间,还可以以一个单独的文件存在,并通过<script>标记符的 src 参数指定包含有 JavaScript 代码的文件。

在编辑工具中输入下面的代码,并保存为网页文件(扩展名为. html 或. htm)。

【例 5-18】 请参看资源包中的例 5-18。

JavaScript 代码可以直接放置在<script>和</script>之间,这部分代码在网页打开时,会按前后顺序依次被自动执行,如示例 5-17 中介绍的 3 个 JavaScript 示例程序。

JavaScript 代码还可以放置在函数中,这时的代码不会被自动执行,而是需要通过事件来激活这部分代码。

在编辑工具中输入下面的代码,并保存为网页文件。

【例 5-19】 请参看资源包中的例 5-19。

可以看到,这个网页中的 JavaScript 代码都是放在 function 这个 JavaScript 关键字中,其表示定义一个函数,在其后面的名字(js_click)就是函数名,函数名后必须跟一个括号对(用于说明传入的参数,如果没有参数,则括号对中间为空)。

为了能够触发这个函数,在网页正文的<input>标记符中多了一个 onclick 参数,其值指明了要执行的函数名(即在<script>标记符中定义的函数 js_click()),这样当使用者单击<input>标记符定义的按钮时,就会执行对应的 JavaScript 函数,这种方法就是事件驱动方式。

在函数 js_click()中只包含一条语句,即 alert("大家好"),其中 alert()是一个系统函数,其作用是弹出一个对话框,此函数的参数表示弹出的对话框中所显示的内容。

5.3.2 JavaScript 变量与数据类型

在任何一个编程语言中都需要存储数据,这些数据一般都存储在变量中。变量顾名思义就是可以变化的量,其中的内容在程序运行过程中是可以发生变化的,因此变量一般是作为运行的中间过程或结果来存储数据的。比如循环中的计数值就要存储在一个变量中,或是存储用户接收的输入值,也要使用变量。

JavaScript 的变量在使用之前,一般要先进行声明,即在使用变量之前,指出变量的名称。JavaScript 可以在变量声明时指定数据类型,也可以通过赋值来指定变量的类型。

下面的语句声明了两个未定义类型变量。

```
var i,j;
i = 1;
j = "Hello, JavaScript";
```

赋值后变量 i 是数字类型,而变量 j 为字符串型。

JavaScript 变量有一套命名的规范,其要求如下。

(1) 变量的第一个字符必须是一个 ASCII 字母(大小写均可),或一个下划线(“_”)。注意第一个字符不能是数字。

(2) 后续的字符必须是字母、数字或下划线。

(3) 变量对大小写敏感,即大写的变量 A 和小写的变量 a 是两个不同的变量。

(4) 变量名称不能与保留字一样,如不能使用 var 或是 function 等。

JavaScript 的保留字如表 5-4 所示。

表 5-4 JavaScript 的保留字

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	false	instanceof	throw	while
debugger	finally	new	true	with
default	for	null	try	

JavaScript 还为将来的扩展提前预留了一些保留字,如表 5-5 所示。

此外,有些对象或系统函数的名字最好也避免使用,以免出现问题,如 String 或 parseInt 等。如果变量没有声明而是直接赋值,这种情况浏览器一般也支持,但建议还是先声明再使用。

变量在声明时一般要指定数据类型,在 JavaScript 中,有 3 种主数据类型、两种复合数据类型和两种特殊数据类型。

表 5-5 JavaScript 扩展的保留字

abstract	double	goto	native	static
boolean	enum	implements	package	super
byte	export	import	private	synchronized
char	extends	int	protected	throws
class	final	interface	public	transient
const	float	long	short	volatile

主数据类型主要有 3 种:字符串类型、数字类型和 Boolean(逻辑)类型;复合数据类型是对象和数组;特殊数据类型是 Null(空)类型和未定义(var,也称为万能型)类型。其中对象和数组将在后面的小节里讲到。

1. 字符串类型

字符串值是一个由零或多个 Unicode 字符(包括字母、数字和标点符号)组成的,其一般表示 JavaScript 中的文本。字符串在使用时必须使用单引号或双引号将其括起来,被单引号括起的字符串内可以包含双引号,被双引号引起的字符串内也可以包含单引号。

2. 数字类型

JavaScript 没有区别整数值和浮点值(包含小数位的数字),统一称为数字类型,一般是通过值来进行判断,例如数字 1 被认为是整数,而包含有小数点的数字 1.0 被认为是浮点数。

3. Boolean(逻辑)类型

Boolean 类型一般作为逻辑判断使用,其只有两个值,一个是真(true),另一个是假(false)。

4. Null(空)类型

此类型只有一个值,即 null(空)。其作用比较少,比如给一个变量赋值为 null 值,就表示清除此变量中的内容。

5. 未定义类型

这个类型表示变量没有明确的类型,需要在赋值后才能定义类型。

5.3.3 JavaScript 运算符与表达式

为了能够进行算术和逻辑计算或是进行字符串操作,JavaScript 提供了多种运算符。下面简要介绍其中比较重要的几个。

1. 算术运算符

算术运算符用于执行变量或值之间的算术运算,主要完成加、减、乘、除等运算。算术运算符如表 5-6 所示。

2. 赋值运算符

赋值运算符用于给 JavaScript 变量赋值,如表 5-7 所示。

表 5-6 算术运算符

运算符	描述	例子	结果
+	加	x=y+2	x=7
-	减	x=y-2	x=3
*	乘	x=y*2	x=10
/	除	x=y/2	x=2.5
%	求余数(保留整数)	x=y%2	x=1
++	累加	x=++y	x=6
--	递减	x=--y	x=4

表 5-7 赋值运算符

运算符	例子	等价于
=	x=y	
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

3. 逻辑运算符

逻辑运算符一般用于语句的判断，返回逻辑类型的值，其运算符如表 5-8 所示。

表 5-8　逻辑运算符

运算符	描述	例子	结　　果
==	等于	a==b	如果 a 和 b 的值相同则返回 true(真)；否则返回 false(假)
!=	不等于	a!=b	如果 a 和 b 的值不相同则返回 true(真)；否则返回 false(假)
>	大于	a>b	a 大于 b 则返回 true(真)；否则返回 false(假)
>=	大于等于	a>=b	a 大于等于 b 则返回 true(真)；否则返回 false(假)
<	小于	a<b	a 小于 b 则返回 true(真)；否则返回 false(假)
<=	小于等于	a<=b	a 小于等于 b 则返回 true(真)；否则返回 false(假)

4. 字符串运算符

字符串运算符只有一个，即加号(“+”)，其用于把文本值或字符串变量前后连接起来。如果是其他类型的值与字符串类型的值使用加号连接在一起，则结果是字符串类型的值。

5.3.4 JavaScript 程序结构

同其他的高级程序语言一样，JavaScript 提供了 3 种程序结构，分别是顺序结构、分支结构和循环结构。

1. 顺序结构

顺序结构最简单，其执行方法是从上到下，中间没有其他的分支线路，图 5-1 所示的“例 5-1”的程序结构就是一个顺序结构。

2. 分支结构

分支结构就是在程序的执行过程中，通过逻辑判断，在多条备选执行线路中，选择出一条来执行。JavaScript 中的分支语句主要有以下几种。

(1) if 语句。

【基本语法】

```
if (条件) {
    条件成立时执行的代码
}
```

其中“条件”是一个逻辑表达式，当值为真时，就执行花括号对(“{”和“}”)中的代码。

【例 5-20】 请参看资源包中的例 5-20。

这段代码在用户登录或注册等界面中经常被使用，通过 JavaScript 代码的判断，可以在页面被提交给服务器前就进行最基本的内容判断，这样就节省了网络带宽和等待的时间。

本示例中的 JavaScript 代码是放在 check()函数中的，由“提交”按钮中的 onclick(鼠标单击)事件执行，即当用户单击“提交”按钮时执行 check()函数。由于 check()函数根

据判断会返回不同的值（真值 true 表示继续提交，假值 false 表示停止提交），这时的 onclick 事件也会执行对应的操作，因此 onclick 事件的值要写为“return check()；”。

在 check()函数中首先判断页面中的用户名输入框中的值是否为空，即等于空，用户名输入框使用页面对象 document 下的属性值，即 document. form1. username. value，如果等于空值，则使用 alert()函数弹出一个提示对话框，并返回 false 值。密码框的判断也是相同的。

（2） if...else 语句。


【基本语法】

```
if (条件){
 条件成立时执行此代码
}
else{
 条件不成立时执行此代码
}
```

if...else 语句要比 if 语句多出一个可选分支，在 if 语句中，如果 if 后面的判断条件为真，则直接执行 if 语句段后面的内容，而当 if 后面的判断条件为假时，转而执行 else 中的语句内容。

【例 5-21】　请参看资源包中的例 5-21。

本例的代码也是使用 onclick 事件驱动执行 check()函数中的 JavaScript 代码。代码中使用了 if...else 语句，在 if 语句后面判断复选框是否被选择，即通过判断 document 对象下的 form1 表单中的 enjoy 复选框的 checked 属性（document. form1. enjoy. checked）的值是否为真，来确定用户是否选择了这个复选框。如果条件为真，则执行 if 后花括号中的语句；如果条件为假，则执行 else 后花括号中的语句。

3. 循环结构

当重复执行同一段代码或是进行遍历数组等操作时就要用到循环语句。循环语句也是程序中最常用的结构之一。

（1） for...in 循环。


【基本语法】

```
for(变量 in 对象或数组){
    在此执行代码
}
```

for...in 循环是 for 循环的一种变形，其主要用于遍历数组或者对象的属性，即对数组或者对象的属性进行循环操作。

【例 5-22】　请参看资源包中的例 5-22。

在这段代码中，首先定义了一个循环使用的变量 x，然后定义了数组对象，并依次对每一个数组元素赋值，最后使用 for...in 循环遍历数组并显示出来。

(2) while 循环。



【基本语法】

```
while(判断条件){
    需执行的代码
}
```

while 循环和 for 循环相似,都可以重复执行一段代码,但 while 循环后的花括号中只有一个继续执行循环条件的判断,因此 while 循环不但可以进行数字表达式的判断,也可以进行其他类型表达式的判断。

【例 5-23】 请参看资源包中的例 5-23。

在本例的代码中,前面先是定义了变量 x 和一个数组,并对数组进行了赋值,然后对遍历数组用的变量 x 进行了初始化,在 while 循环中依次检查每一个数组的值,直到值是 black 为止循环才结束,在循环中进行了内容的显示,同时对变量 x 进行了增量操作,这里需要注意,如果在循环中忘了对 x 进行增量,则循环将变成死循环。

5.3.5 JavaScript 函数

JavaScript 程序主要以函数为单位进行编程设计,函数一般是由事件来驱动,或者被其他函数调用。使用函数的最初目的是减少编程工作量,将可重复使用的代码块放在函数中以方便调用,而现在则加入了使源代码阅读更清楚的目的。

JavaScript 程序中主要提供了两种函数:一种是系统自带的函数;另一种是由用户自定义的函数。

JavaScript 程序中自定义函数的创建方法非常简单。



【基本语法】

```
function 函数名(var1,var2,...,varX){
    代码块...
}
```

其中 function 为关键字,而"函数名"位置则使用与声明变量一样的规则给函数起一个本网页中唯一的名字,再后面是括号对,在括号对中是传入函数中的参数,如果有多个参数,则使用逗号分隔;如果函数没有参数,则只保留括号对。然后是由花括号包含的代码块。

JavaScript 程序中函数的调用一般有两种情况:①通过事件驱动来调用,如按钮对象的鼠标单击事件;②由其他函数来调用。

调用的方法是在需要调用时直接写上函数的名称,如果有参数,把参数放在调用的函数名后的括号对中,多个参数之间使用逗号分隔;如果没有参数,则也要增加一个括号对在函数名的后面。如果调用的函数有返回值,则需要在被调用的函数名前加上一个接收变量和一个等号。

有些函数经过运算后会有返回值，在函数中可以使用 return 语句。一般在 return 语句的后面会跟一个变量或常量。如果函数中包含多条 return 语句，则只有第一个被执行的 return 语句有效，另一个 return 语句只能返回一个值。

为了方便开发，系统不但允许用户自己定义函数，还提供给开发者丰富的编程资源，这些资源都是以函数的形式出现，因此也称其为系统函数。这些函数一共可分为五类：常规函数、数组函数、日期函数、数学函数和字符串函数。其中除了常规函数，其他函数都要和对象一起使用。

在系统函数中，最常使用的就是消息对话框函数了，它不但用于网页运行时的一些提示性显示，有时还可以应用在调试过程中。消息对话框函数主要有 3 种弹出对话框，分别是使用 alert()、confirm()及 prompt()方法调用。

1. alert()对话框

**【基本语法】**

```
alert(字符串);
```

其主要功能是显示提示信息，在弹出的对话框中，会将 alert()函数的参数作为提示内容显示在对话框上，并在下面显示一个“确定”按钮。

2. confirm()对话框

**【基本语法】**

```
ret = confirm(字符串);
```

其主要功能是显示一个可以选择的提示信息，在对话框中会将 confirm()函数的参数作为提示信息显示出来，并在下面显示出一个“确定”按钮和一个“取消”按钮。本函数会根据用户的选择返回不同的值，如果单击“确定”按钮将返回 true；否则返回 false。

3. prompt()对话框

**【基本语法】**

```
ret =  prompt(提示字符串,默认输入框内的字符串);
```

其主要功能是显示一个可以输入内容的对话框，在对话框中会将 prompt()函数的第一个参数作为提示信息显示出来，下面是一个文本输入框，其默认值就是 prompt()函数的第二个参数(一般这个值是空，即"　")，再下面是一个“确定”按钮。当用户在输入框中输入内容并单击“确定”按钮后，该对话框会关闭，并将用户输入的字符串返回给调用者。

【例 5-24】 请参看资源包中的例 5-24。

5.3.6　JavaScript 对象

对象是 JavaScript 程序中比较重要的概念之一，很多实用性的功能必须要依靠对象才能实现。

JavaScript 是面向对象的编程语言(OOP),因此其提供了非常强大的对象功能。JavaScript 提供了很多已经定义好的内部对象,也允许用户自定义对象,但由于本书定位于入门级水平,因此只讲解内部对象。

对象可以看成是一种特殊的数据,每一个对象都包括众多的属性和方法。属性是指与对象有关的值,其表现形式一般是变量;方法指对象可以执行的行为,其表现形式一般是函数。

下面的代码显示了一个字符串,并通过调用对象的属性和方法,显示了字符串的长度和全部变成大写后的结果。

```
<html><head>
<title>例 5-25</title>
<script language = "javascript">
    var txt = "Hello World!";
    document.write("<p>原始字符串是: " + txt);
    document.write("<p>字符串的长度是: " + txt.length);
    document.write("<p>全部变成大写后的字符串是: " + txt.toUpperCase());
</script></head>
<body></body></html>
```

在 JavaScript 代码开始处首先是定义了一个包含有大小写字母的字符串,然后分别调用了 3 次 document 对象中的 write()函数(即 document 对象的 write()方法),第一次显示了字符串本身;第二次调用了字符串对象中的 length 属性,显示了字符串的长度;第三次使用了字符串对象的 toUpperCase()方法显示了转换成大写的字符串。

【例 5-25】 请参看资源包中的例 5-25。

5.3.7 JavaScript 事件响应

JavaScript 程序和网页页面之间的交互一般是通过触发浏览器界面中的控件,进而引发事件来处理的。

网页中的每个元素都可以产生某些可以触发 JavaScript 函数的行为,这些行为可以被 JavaScript 程序侦测到,称其为事件。例如,用户在页面中单击某按钮,则会产生一个鼠标单击(onClick)事件,并调用之前在按钮的 HTML 标记符上关联的 JavaScript 函数,完成对事件的处理。

如果需要对某一事件进行处理,需要进行两部分的设置。

(1) 在相应网页页面的元素标记符中加入事件属性,并设置事件属性为一个 JavaScript 函数。

(2) 在 JavaScript 程序中编写一个函数处理对应的事件。

JavaScript 程序支持的事件如表 5-9 所示。

表 5-9 JavaScript 程序支持的事件

事 件 名 称	功 能 解 释
onabort	图像加载被中断
onblur	元素失去焦点
onchange	用户改变域的内容

续表

事件名称	功能解释
onclick	单击某个对象
ondblclick	双击某个对象
onerror	当加载文档或图像时发生某个错误
onfocus	元素获得焦点
onkeydown	某个键盘的键被按下
onkeypress	某个键盘的键被按下或按住
onkeyup	某个键盘的键被松开
onload	某个页面或图像被完成加载
onmousedown	某个鼠标按键被按下
onmousemove	鼠标被移动
onmouseout	鼠标从某元素移开
onmouseover	鼠标被移到某元素之上
onmouseup	某个鼠标按键被松开
onreset	“重置”按钮被单击
onresize	窗口或框架被调整尺寸
onselect	文本被选定
onsubmit	“提交”按钮被单击
onunload	用户退出页面

鼠标事件是 JavaScript 程序中最常用的事件之一，其功能是响应鼠标的各种操作，主要包括以下几个常用事件。

1. 单击事件(Onclick)

本事件最常用，当单击时产生，一般与按钮元件配合使用，本章中很多实例都使用了该事件。

2. 鼠标指向事件(onMouseOver)

当鼠标指向某一目标区域时，会触发此事件。

3. 鼠标移开事件(onMouseOut)

当鼠标离开某一目标区域时，会触发此事件。鼠标移开事件和鼠标指向事件经常配合使用。

菜单的设计一般是使用块和表格共同完成的，每一个菜单都定义在一个块(<div>和</div>)中，并对这个块增加鼠标指向和移开事件的处理函数，如同下面的语句。

```
<div onmouseover = viewX() onmouseout = hideX() class = cur >
```

鼠标事件的处理函数的内容非常简单，只是设置相应对象的 style. visibility 属性值即可，当值是 visible 时表示菜单可见，当值是空值(即" ")时表示菜单不可见。

【例 5-26】 请参看资源包中的例 5-26。

在本程序开始处首先使用 CSS(层叠样式表)定义了 4 个样式，其中. menu 样式(其内容是{ VISIBILITY: hidden; })设置了不可见属性，当其应用在定义菜单的块上时，则将

菜单部分隐藏。

在每一个菜单项中还使用了单击事件,以便完成颜色、背景色和字体大小的改变。



本章小结

本章主要介绍了网站设计的方法,对网站规划、制作流程、需求分析进行了详细的介绍。然后介绍了网站总体设计的内容,对布局设计、色彩使用和页面设计进行了讲解,其中最重要的一点就是网站的风格一定要统一。

基于 HTML 的页面其美化效果比较差,使用 CSS 技术可以弥补这个缺陷,而 JavaScript 可以让网页提高交互性能,增加显示效果。CSS 和 JavaScript 都是现今网站开发时必不可少的技术,它们的功能非常强大,本章只用了两节内容对其进行了简单介绍,更详细的内容请读者参考其他相关书籍或网站。



思考与练习

1. 根据本章的介绍,为某学校设计一份校园网网站建设方案。
2. 了解 Ajax 技术,并将其应用到读者开发的网站中。

第 6 章

网站快速开发技术

网站快速开发技术是指使用 CMS(Content Management System,内容管理系统)等技术通过现有的网站模板快速生成一个网站,然后经过修改就可以达到快速部署的网站开发技术。

6.1 内容管理系统

内容管理系统(CMS)是进行网站快速开发的主要技术,它是一种位于 Web 前端(Web 服务器)和后端办公系统或流程(内容创作、编辑)之间的软件系统。主要是以信息共享为目的,面向海量信息处理,集信息数字化、分布存储、管理、传播、查询于一体的管理平台。

它分离了内容的管理和设计,在多数的 CMS 系统中,页面设计存储在模板里,而内容存储在数据库或独立的文件中,当一个用户请求页面时,页面外观和页面的内容联合生成一个标准的 HTML 页面供用户使用。

6.1.1 内容管理系统概述

内容管理系统采用数据库技术,把数据库中的信息按照规则预先自动生成 HTML 页面,或者利用动态网页生成技术,在实时交互的过程中动态产生网页。系统包括信息采集、整理、分类、审核、发布和管理的全过程,具备完善的信息管理和发布管理功能。利用内容管理系统,可以随时方便地提交发布的信息而无须掌握复杂的技术。

目前主流的 CMS 概念,是以文章系统为核心,包含内容模型自定义、内容采集加工发布、内容评论、内容检索、广告管理、调查管理、留言管理等各种通用功能,面向各种内容管理需要的 CMS。而"内容"包括文件、表格、图片或数据库中的数据甚至视频等一切网站中需要发布的内容。

内容管理系统主要经历了以下 3 个发展阶段。

第一阶段,完全手动型。最为原始的内容管理模式,在这个阶段使用者设定好背景

色、一系列的字体和 CSS,同时在最终的网页上手动完成相应的链接。

第二阶段,数据库支持型。在内容增加到一定程度时,通过使用数据库来存储大量的信息,在 Web 服务器上使用如 ASP、PHP、JSP 这样的程序从数据库中写入和取出相应的数据。

第三阶段,页面自动生成型。技术人员通过开发了自定义的标签,将页面的模板独立出来,用数据库中的数据与标签内指定的数据结合起来,动态生成用户看到的最终静态页面。

CMS 技术的软件有很多,在选择 CMS 解决方案时需要考虑以下几条原则。

(1) 良好的架构和可扩展性、方便维护和管理。内容管理系统需要基于优良强健的体系架构,遵从开放标准,易于与其他应用相集成和功能扩展,需要提供方便的管理维护功能或工具,并可以快速部署。

(2) 易用性、灵活性、可用性和安全性。内容管理平台从界面交互友好性、需求符合度、功能应用灵活性到扩展选件的多样化等方面需要表现出良好的易用性和可用性。随着内容应用环境进一步复杂和开放,对系统安全保障机制也提出了高要求。

(3) 符合需求的系统功能。不同的内容管理系统提供的功能和模块存在差异,在挑选内容管理系统时,可能会因为某项特殊的需求而必须要使用某套系统,或对于同样功能的 CMS,但其运作方式相差甚大,需要从中挑选出合适的 CMS。

(4) 表现和内容分离,用户体验和内容质量的和谐统一。与系统管理和内容业务分离相类似,内容表现和内容本身需要尽可能独立;无须内容生产人员关注过多的内容表现形式的制作,同时,页面设计和内容创建应当符合设计人员和编辑人员的工作习惯,支持专业设计工具,提供符合常用操作习惯的、易用的工作界面,使得工作人员和最终用户均获得满意的用户体验,保证提供与完美表现相结合的高质量内容。

(5) 系统低总拥有成本和高价值导向。内容管理产品是一个具有平台性概念的开放产品,面向快速部署和灵活扩展,保证系统整体的高效率和灵活性,降低总拥有成本。在挑选内容管理系统时需要考虑到系统相关的成本费用,包括授权费、服务费、二次开发费等。

6.1.2 基于.NET 技术的 CMS 软件

CMS 软件的种类有很多种,每种软件所基于的技术也不尽相同,比如有基于 Java 类的 TurboCMS、TubroECM 等内容管理系统;基于 PHP 的织梦、帝国等内容管理系统;基于 ASP 的动易、新云等 CMS 系统;基于.NET 的 Kooboo、DotNetNuke 内容管理系统等。

每种 CMS 软件都有自己的特点,在实际快速建设网站中,究竟要选取什么样的 CMS 系统,可参考 6.1.1 小节所提到的 5 条原则。

尽管在实际中,运用基于 PHP 技术的 CMS 软件(如织梦)较多,但由于本教材主要是通过 C#语言讲述 ASP.NET 动态网站程序设计的,因此本章的重点内容将介绍基于.NET 技术的 CMS 软件。

1. DotNetNuke(DNN)

DotNetNuke既可用作简单网站的Web内容管理系统(CMS),也可作为强大的应用程序开发框架,使企业能够在Microsoft Web平台上迅速构建和部署功能丰富的交互式网站和应用程序。

它具有以下特点。

(1) 方便用户。DotNetNuke旨在使用户可以更轻松地管理所有方面的项目,设置了网站向导、帮助图标,并在良好研究基础的用户界面,让全民易于操作。

(2) 强大。DotNetNuke可以支持多个子网站。通过Host账号管理所有子站点,而每个子站点都有独自的管理员,让管理着的任意数量的网站以及每个成员都有其自身的外观和身份。

(3) 功能丰富。DotNetNuke预装了一套内置的工具,提供了强大的软件功能,通过这些工具,网站主机、设计、内容、安全性和成员的选择都可轻松管理和定制。

(4) 支持。DotNetNuke是支持它的核心开发团队以及一个专用于国际社会。通过用户群体、网上论坛、资源门户网站和网络公司的代表,他们专门从事dnn工作,使支持无处不在。

(5) 易于安装。几分钟内就可以安装DotNetNuke。只要按照安装说明,简单地从www.dotnetnuke.com即可下载该软件。

(6) 本地化。DotNetNuke包括一个多语言本地化功能,该功能让管理员可以轻松地将他们的项目和门户变成任何一种语言。

(7) 开放源码。DotNetNuke是免费提供的。

(8) 可扩展。DotNetNuke能够创造出最复杂的内容管理系统,完全符合其内置功能,也使系统管理员能够有效地开展工作,与加载项第三方集会,并可定制工具。网站个性化和功能是无限的。

这款CMS软件可从微软的官网http://www.microsoft.com/web/dotnetnuke/上得到。

2. Rainbow

Rainbow是一个使用Microsoft ASP.NET和C#技术开发的全面的CMS内容管理网站系统。它支持29种语言,内容可以委派给基于角色的组成员管理的会员,后者不需要懂得HTML知识,Rainbow支持两步的审核发布机制。发布的版本包含了75种插件,如电子商店、XML feeds、地图、邮件列表、调查、论坛、文档管理及客户列表等。开发文档中也提供了如何自己设计自定义模块功能的指导。

这款CMS的官方网址是:http://www.rainbowcms.com。

3. NetCMS

NetCMS是基于ASP.NET 2.0开发的网站内容管理系统,程序完全开源,没有任何文件加密,不需要注册任何组件,方便二次开发。

它具有以下特点。

(1) 模板与程序分离,标签调用,支持DIV+CSS,批量设置属性,让模板制作更简单。全新“网站模板与网站程序完全分离”的概念,具有强大的标签加样式的个性化组合,

可自定义标签、自定义表单、JS 管理加 JS 模型(自定义 JS、系统 JS)的灵活应用，支持不同频道、栏目、内容页、专题等应用不同的模板，随时能编辑、修改和更换网站界面，系统集成了类同 Macromedia Dreamweaver 一样简单的可视模板编辑方式，可批量设置属性，模板标签全面支持目前最受欢迎的 DIV+CSS 格式，支持批量绑定模板，完全做到轻松换肤。

(2) 自定义频道。用户可根据自己的需要自定义数据表，自定义字段，从而组合出新的频道，如房产、招聘等，每个频道都可以拥有独立的表，彻底减轻了数据库的压力。自定义字段的内容录入可支持 HTML 可视化编辑，方便前、后台数据录入界面的排版布局，扩展出更多的特殊效果。

(3) 自定义表单。自定义表单功能支持时间限制、用户组权限控制、奖励扣除金币、附件上传、验证码等功能，可轻松实现订单预订、问卷调查等效果。

(4) 自定义标签。基于此功能，可以轻松编写 SQL 语句，实现任意数据任意调用，相关链接可根据关联条件查询。

(5) 支持动态访问和静态发布。整站程序支持全静态 HTML 文件生成，可将站点首页、频道首页、各栏目及每个内容页都生成静态 HTML 文件，这样不仅可以减轻服务器的负载，提高搜索收录率，同时也可以实现内容收费和访问权限控制。有多种生成文件命名形式可供选择，可自定义文件存放路径，可以随心所欲地设置要生成的扩展名，可根据喜好 DIY。动态访问可支持伪静态，增强 URL 友好性，方便搜索引擎的收录。

(6) 自定义菜单、快捷方式，维护变得更轻松。支持用户按自己使用习惯添加、修改菜单、设置快捷方式，让网站维护变得更轻松。

(7) 支持数据库字段替换功能，可在线执行 SQL 语句，具有数据备份和恢复功能。系统具有强大的数据库字段替换功能；强大的在线执行 SQL 语句功能；强大的数据备份和恢复功能，可以在线备份、恢复、压缩数据库。

这款软件的官方网址是：http://www.aspxcms.com。

4. PageAdmin

PageAdmin 网站内容管理系统是一款基于微软 ASP.NET 平台开发，集成内容发布、文章、产品、图片、招聘、留言、自定义模型、采集等功能于一体的企业级网站管理系统。

它具有以下特点。

(1) 可视化的管理网站内容和结构。自由的模块布局和组合设计，灵活的内容调用，强大的自定义功能，所见即所得的编辑器等功能都体现了网站架构的自由和灵活，网站可以做得很简洁，也可以做得很复杂，完全可以根据自己需求来架设。

(2) 自由设计网站风格界面。系统采用 DIV+CSS 结构，遵循国际最新 W3C 网页设计标准，兼容 IE、火狐、Opera 等主流浏览器，结构和设计的完全分离让网站界面想换就换。

(3) 周密的安全策略和攻击防护。对 SQL 注入攻击进行过滤、对密码进行了不可逆加密处理，提供了数据库备份功能、对管理员权限的自由分配，在方方面面保证了系统的安全和稳定。

(4) 降低网站开发和维护成本。通过安装 PageAdmin，任何用户都可以轻松地架设自己需要的网站。在给用户高效、简单建站的同时还减少了总成本。

这款软件的官方网址是：http://www.pageadmin.net/。

5. Umbraco

Umbraco 是一个开源的 CMS 内容管理系统，基于 ASP.NET 建立，使用 MSSQL 存储数据。使用 Umbraco，设计师能创造出有效的 XHTML 标记模板，并且开发人员可以创建任何基于.NET 的模块。

这款软件的官方网址是：http://www.umbraco.org/。

6. Kooboo CMS

Kooboo 是一个基于 ASP.NET 的 CMS 系统，实现面向企业级的内容管理解决方案和快速开发。它是一款由中国人开发的，但走国际路线的 CMS 软件。Kooboo 具有以下主要特性。

(1) 基于角色的用户管理。

(2) 无限制的用户和站点。

(3) 可实现各种验证。

(4) 可进行内容版本控制。

(5) 可进行工作流控制。

(6) 具备布局和内容模板。

这款软件的官方网址是：http://www.kooboo.com。

基于.NET 的 CMS 软件有很多，这里仅列举了其中知名度比较高的 6 款软件。要学会使用它们，首先要学会 ASP.NET。但这些软件多是基于 MVC(Model View Controller，模型—视图—控制器)的 ASP.NET 的 Web 编程模式，这与本书其他章节所讲授的 Web Forms 编程有着截然不同的编程模式，因此，本章仅讲授 CMS 软件入门使用，不做深入的介绍。有兴趣的读者可在先行学习 ASP.NET MVC 和 Razor 语法的基础上深入研究基于.NET 的 CMS 软件。

下面将以 Kooboo CMS 为代表说明基于.NET 的 CMS 软件的使用。

6.1.3 CMS 软件的安装

不同的 CMS 软件的安装是类似的。安装其实就是将一个已经设计好的 ASP.NET 网站架设到本地，然后进行相应的配置即可。

这里以 Kooboo CMS 的安装为例进行说明。Kooboo CMS 有两种安装方式：一种是站点架设的方式；另一种是通过 Web Matrix 3 进行安装。下面分别介绍这两种安装方式。

1. 架设站点方式安装 Kooboo CMS

1) 安装前准备

(1) 操作系统。Microsoft 的 Windows 操作系统。

(2) Web 服务器。安装 IIS 6 以上的版本。

(3) NetFramework。安装有 NetFramework 4。

(4) 安装者身份。具有管理员权限的用户。

2) 安装方法

由于 IIS 6 和 IIS 7 的安装方法略有不同，这里以 IIS 7 安装为例进行说明，使用 IIS 6

的读者可参考官方网站的相应说明。

(1) 从 http://kooboo.codeplex.com 下载最新版本的 Kooboo_CMS 安装包解压到 C:\Kooboo_CMS。

(2) 打开 IIS 7 控制台，创建一个应用程序池。.NET 框架版本选择 .NET Framework v4.0.30319，通道模式选择“集成”模式，命名为 Kooboo_CMS pool，如图 6-1 所示。



图 6-1　创建 Kooboo Pool

(3) 在 IIS 7 控制台创建一个新站点。站点目录指向 C:\Kooboo_CMS，应用程序池使用刚刚创建的 Kooboo_CMS pool，如图 6-2 所示。

(4) 设置站点的目录权限。Kooboo CMS 要求当前的站点运行用户具有对 Cms_Data 目录的读写权限。进入 C:\Kooboo_CMS 中，选择 CMS_Data 文件夹，在其安全的属性中，设置 Everyone 具有完全控件权限，如图 6-3 所示。



图 6-2　创建 Kooboo 网站



图 6-3　设置 CMS_Data 权限

2. 通过 Web Matrix 3 进行安装

Web Matrix 3 是微软公司提供的一款轻量级的完全免费的网站开发、部署工具。它里面集成了许多知名的 CMS 建站软件。

1）下载安装 Web Matrix 3

可从 http://www.microsoft.com/web/webmatrix/下载安装 Web Matrix 3。安装 Web Matrix 3 会同时安装 SQL Server Compact、IIS 8 Express 和.NET framework 4。如果这些之前都没有安装过，Web Matrix 3 会自动进行安装，但安装时间也会很长。安装成功后进入的界面如图 6-4 所示。



图 6-4　Web Matrix 3 启动界面

2）安装 Kooboo CMS

进入 Web Matrix 3，选择“新建”→“应用程序库”菜单命令，如图 6-5 所示。用户在这里可选择多种 CMS 软件进行安装。



图 6-5　Web Matrix 3 新建应用程序库

通过搜索 Kooboo 可以找到要安装的 Kooboo CMS。选择 Kooboo CMS，按照向导单击“下一步”按钮进行安装即可，如图 6-6 所示。图 6-7 是 Kooboo CMS 安装的第一步，在图 6-8 中需要用户同意使用协议（单击“我接受”按钮，安装才能顺利完成），使用 Web Matrix 3 安装 Kooboo CMS，系统会首先安装 ASP.NET MVC 4.0。

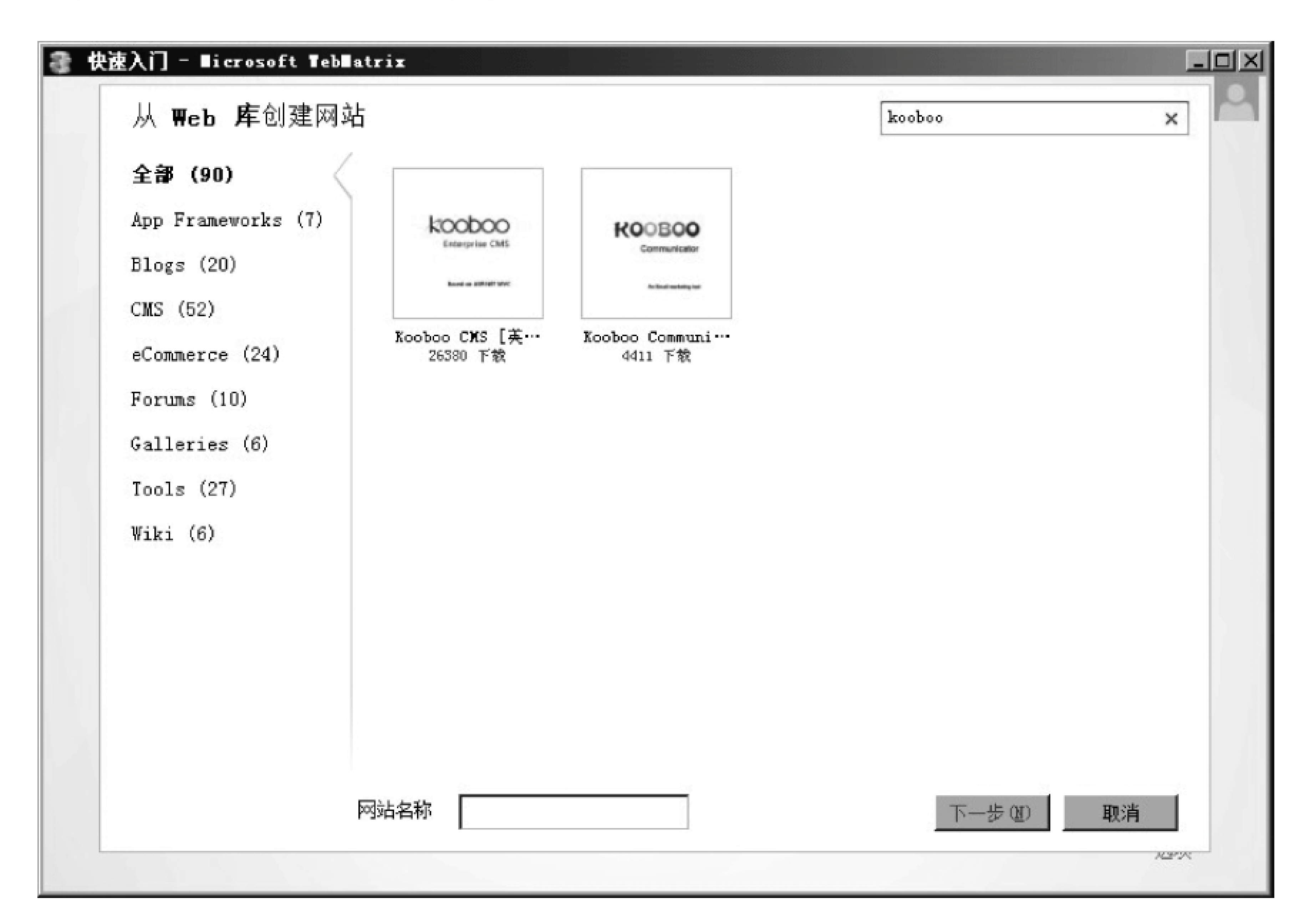


图 6-6 查找 Kooboo CMS



图 6-7 安装 Kooboo CMS

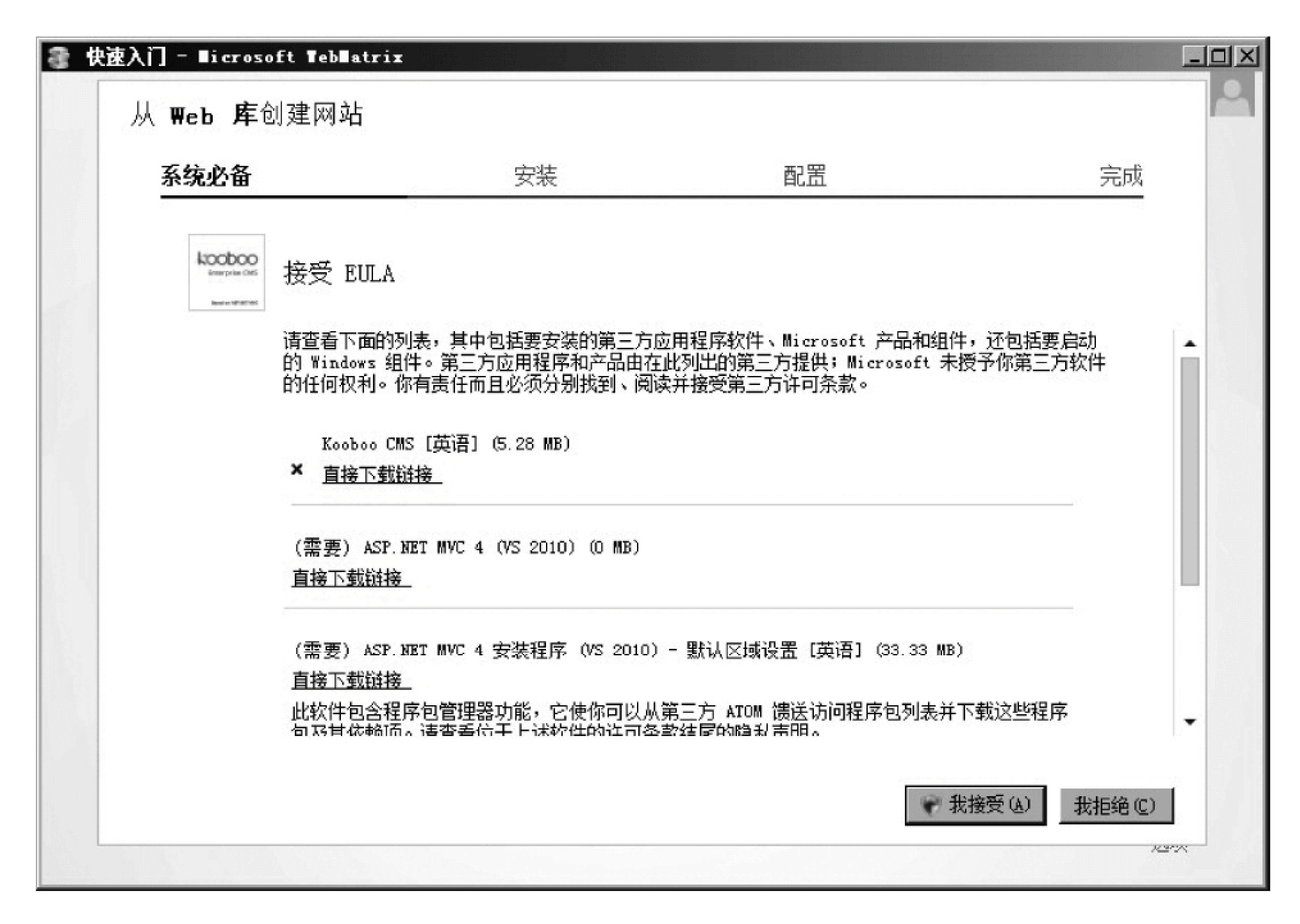


图 6-8　接受 Kooboo CMS 协议

6.2　Razor 视图引擎

Razor 视图引擎是 ASP.NET MVC 编程的基础。Razor 语法简洁，编写起来十分方便。前面提到了很多基于.NET 的 CMS 系统均采用了 ASP.NET MVC 编程模式，因此，也支持 Razor 语法。故这里先对 Razor 语法做简要的介绍。

6.2.1　Razor 语法概述

Razor 是伴随 ASP.NET MVC 编程模式而出现的，它包含了模板引擎和动态编译两部分。这里只介绍模板引擎的语法。Razor 可以在 VB.NET 和 C# 中使用，分别对应两种文件类型，即 cshtml 和 vbhtml。在 Web Matrix 3 中创建 Razor 文件的方法如下。

(1) 启动 Web Matrix 3，选择“新建”→“从模板库创建站点”→“空白网站”菜单命令。

(2) 单击“新建文件”，弹出图 6-9 所示的对话框。

(3) 此时选择 cshtml 文件类型，此时建立的 Razor 页的代码如图 6-10 所示。

(4) 如要运行此页面，可以在左侧的网站目录中要运行的页面上右击，在弹出快捷菜单中选择“在浏览器中启动”命令，即可运行此网页。

图 6-10 中显示的@字符被定义为 Razor 服务器代码块的标识符，后面使用大括号{}表示服务器代码(作用域)，这就和 Web form 中使用<%%>写服务器代码是同样的道理。



图 6-9 Web Matrix 3 创建 Razor 页面

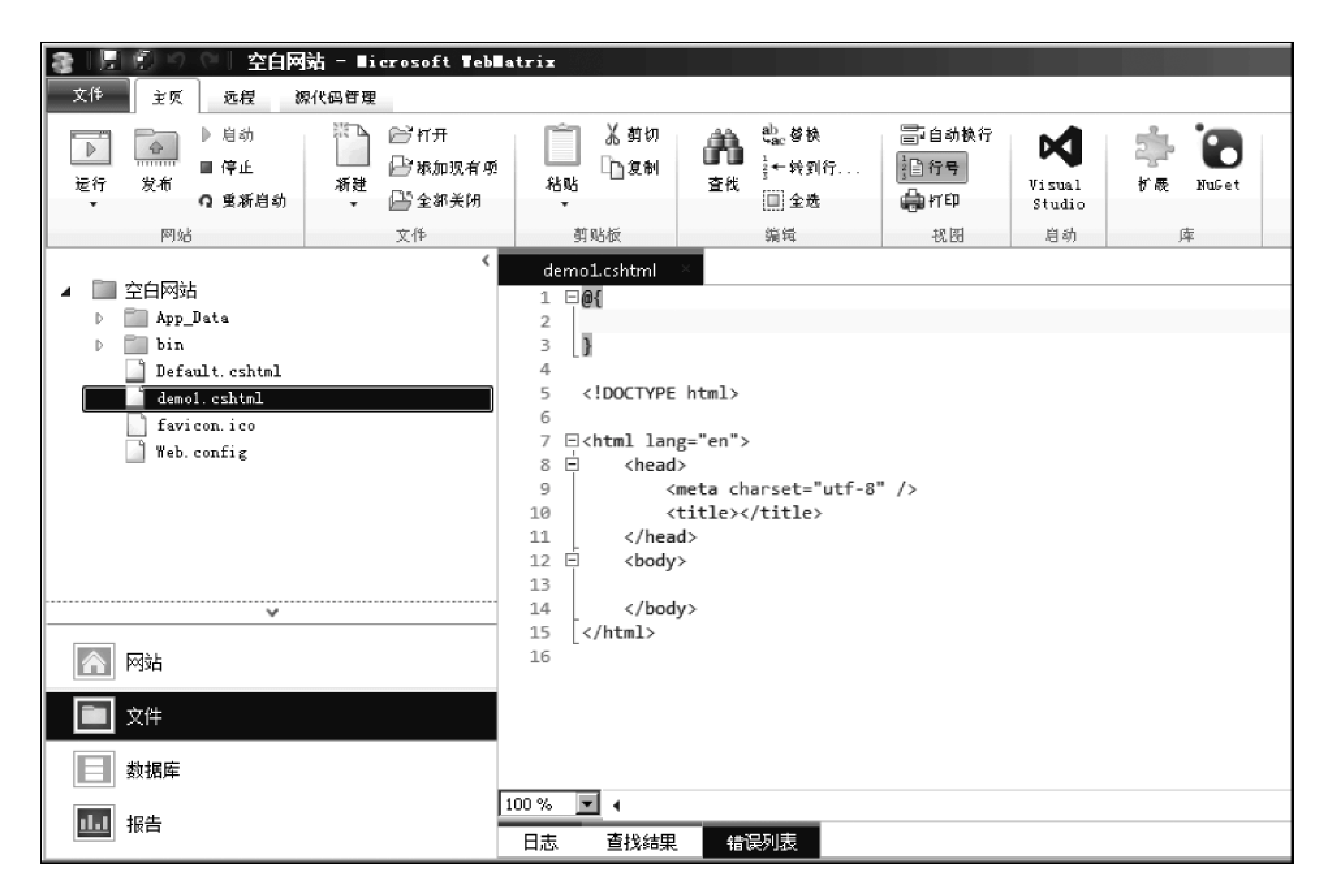


图 6-10 Razor 页面的代码

Razor 中可用 var 定义变量，使用 var 定义变量不需要指明数据类型，在运行时，变量会根据赋的数值自动设定数据类型。

在 Razor 中写 HTML 代码或在 HTML 代码中写 Razor 语句都是可以的。

（1）在作用域内如果是以 HTML 标签开始则视为文本输出。

(2) 如果要输出@,则使用@@。

(3) 如果要输出非 HTML 标签和非 Razor 语句的代码,则用@:。

(4) @:后面加上@就是表示 Razor 语句的变量。

【例 6-1】 示例见资源包中的 6-1. cshtml。

【示例说明】

(1) @:字符的使用。

(2) Var:定义变量的方法。

(3) Razor 的变量可以直接当作网页的标题(title)。

Razor 作用域里面是服务器代码,因此可使用服务器代码的注释,注释有//、/*和*/,分别表示单行注释和多行注释。另外,Razor 注释还可以使用@* *@。

【例 6-2】 示例见资源包中的 6-2. cshtml。

【示例说明】

Razor 使用注释的几种方式。

在 Razor 中可以使用@href 进行超链接的设置。

【例 6-3】 示例见资源包中的 6-3 文件夹中的 index. cshtml、about. cshtml 和 content. cshtml。

【示例说明】

href="@("about. cshtml")"这种形式表示超链接。

6.2.2 Razor 中的变量和数组

1. Razor 中的变量

Razor 中使用 var 定义变量不需要指明类型,变量的类型会根据数据自动设定。常用的数据类型见表 6-1。

表 6-1 Razor 常用的数据类型

类型	描述	实 例
int	整数	103、12、5168
float	浮点数	3. 14、3. 4e38
decimal	小数	1037. 196543
bool	逻辑值	true、false
string	字符串值	"Hello W3chtml" "Bill"

例如:

```
var a = 123.34;
var b = "1234";
var c = "true";
```

2. Razor 中的数组

Razor 中的数组可采用下面的形式进行定义。

```
Var animals = new string[ ]var animals = new string[ ]{"fish","dog","cat","cow","duck"};
```

使用 animals 数组中的元素时可采用 animals[2]形式。



【小提示】

数组的编号从 0 开始，上面的 animals[2]的值是"cow"。

如果要访问数组中的每一个元素，可用 foreach 循环遍历数组。

【例 6-4】 示例见资源包中的 6-4. cshtml。

【示例说明】

(1) Razor 使用数组的方法。

(2) 通过 foreach 循环遍历数组每一个元素的方法。

数组也可以 new list<string>()方式进行声明，然后在运行中通过 Remove 或 Add 方法减少或增加数组元素。

【例 6-5】 示例见资源包中的 6-5. cshtml。

【示例说明】

(1) Razor 使用数组的方法。

(2) 通过 Remove 减少数组元素。

(3) 通过 Add 方法增加数组元素。

3. Razor 变量类型的转换和判定

Razor 中可通过 As 系列函数进行数据类型的转换，通过 Is 系列函数进行数据类型的判定。

【例 6-6】 示例见资源包中的 6-6. cshtml。

本示例运行结果如图 6-11 所示。

值	IsEmpty()	IsBool()/AsBool()	IsInt()/AsInt()	IsDateTime()/AsDateTime()	IsDecimal()/AsDecimal()	IsFloat()/AsFloat()
	True	False/False	False/0	False/0001/1/1 0:00:00	False/0	False/0
true	False	True/True	False/0	False/0001/1/1 0:00:00	False/0	False/0
1	False	False/False	True/1	False/0001/1/1 0:00:00	True/1	True/1
1234	False	False/False	True/1234	False/0001/1/1 0:00:00	True/1234	True/1234
123.987	False	False/False	False/0	False/0001/1/1 0:00:00	True/123.987	True/123.987
2015/5/1	False	False/False	False/0	True/2015/5/1 0:00:00	False/0	False/0
	True	False/False	False/0	False/0001/1/1 0:00:00	False/0	False/0

图 6-11 As 系列和 Is 系列函数演示结果

【示例说明】

(1) Is 系列函数就是判定一个变量的值是否符合某种数据类型，返回值是 bool 型(即要么是 true 要么是 false)。

(2) As 系列函数则是将一个变量的值自动转换为指定的数据类型，注意不同类型之间的转换，转换后的结果不一定是正确的。

(3) 注意" "字符串与 null 的判定和转换结果是否相同。

6.2.3　Razor 中的自定义函数

Razor 中的自定义函数可通过@helper 完成。如下面代码表示的一个自定义的函数 sum,其作用是计算传递过来的两个整型参数之和。

```
@helper sum(int i,int j)
{
    var result = i + j;
    @result;
}
```

调用时,采用@sum(参数 1,参数 2)的形式就行了。

【例 6-7】　自定义函数示例见资源包中的 6-7.cshtml。

在实际使用过程中,通常是专门建立一个用于存储函数的文件(文件类型也是 *.cshtml),并将此文件保存到网站的 App_Code 文件夹中,然后通过“文件名.函数名”的方式调用即可(存放函数名的文件就类似于类名)。

【例 6-8】　示例见资源包中的 6-8.cshtml 及 App_Code 文件夹内的 define.cshtml 文件。

【示例说明】

(1) define.cshtml 只存储自定义函数。

(2) 本例有两个自定义函数,一个是 sum(求两个数之和);另一个是 multiplicate(求两个数之积)。

(3) 调用的形式:@define.sum(2,5)。

如果网页中有多个超链接,也可以利用@helper 的自定义函数形式来设置。

【例 6-9】　本例见资源包中的文件夹 6-9 中的 index.cshtml、content.cshtml 和 about.cshtml。

【示例说明】

(1) NavigatorItem 是个自定义函数,其作用是接收两个参数 name 和 item,item 参数用于设置超链接的文件名,name 参数用于超链接显示的文字。

(2) 调用的形式:@NavigatorItem("网站内容","content.cshtml")。

(3) Ltem 参数是一个网页文件的名称。

6.2.4　Razor 中的布局

Razor 中的布局(layout)的概念类似于 Web Form 编程模式中的母版页。作为布局的网页,如同母版页,而使用布局的网页就是应用母版页的网页。

在网页中使用布局可以统一网站风格:对于作为布局的页面,其标题、CSS、网页中的静态内容等都会成为使用此布局网页的元素。这样,多个网页就达到了一致的风格。

作为布局的页面,其名称必须以“_”开始,其页面代码中一定要添加@RenderBody()语句(此语句用来指示在此显示使用布局的页面)。

作为使用上述布局页的网页,需要在开始的@{　}中添加 Layout="布局页名称",

这样使用布局页的页面内容就出现在作为布局页的@RenderBody()处。

【例 6-10】 本例见资源包中的文件夹 6-10 中的_Layout. cshtml、index. cshtml 和 about. cshtml。

【示例说明】

(1) _Layout. cshtml 是作为布局的页面，作为布局页面的网页文件，是不能像普通网页一样通过浏览器进行浏览的。

(2) index. cshtml 和 about. cshtml 是使用布局的页面。请注意 Layout＝"_Layout. cshtml"语句的使用。

(3) 注意在网页运行时，虽然 index. cshtml 和 about. cshtml 有自己的网页 title，但在显示时，都是_Layout. cshtml 的标题，而不是自己的标题。

(4) 注意使用布局的网页内容出现的位置和@RenderBody()语句的关系。

从上面的示例可以看到，使用布局的网页自身内容只能显示在作为布局页的@RenderBody()的位置处，而其他位置则显示的是布局页中的内容。有时可能需要在使用布局页的页面中属于布局页内容中的某个指定位置显示自己特有的内容，这时可采用@RenderSection 来完成此功能。

【例 6-11】 本例见资源包中的文件夹 6-11 中的_Layout. cshtml、index. cshtml 和 about. cshtml。

【示例说明】

(1) _Layout. cshtml 是作为布局的页面。其中的@RenderSection("subtitle")表明此位置将显示使用布局的页面中的自定义节 subtitle 中的内容。本例是在此显示网页的标题。第二个@RenderSection("subindex")的道理与此类似。

(2) @RenderSection 中的 subtitle 和 subindex 是自定义节的名称，需要在使用布局页的@{ }中进行定义。定义的方法类似于下面的代码。

```
@section subtitle
{
    <title>这是使用 section 方法生成的网页标题</title>
}
```

(3) 由于 index. cshtml 和 about. cshtml 是共用_Layout. cshtml 模板的，而 about. cshtml 中并未定义 subtitle 和 subindex 节，请观察 about. cshtml 能否正常运行。

有时需要在一个页面中展示另一个页面的内容，此时可以用@RenderPage 语句完成此操作。

【例 6-12】 本例见资源包中的文件夹 6-12 中的_Layout. cshtml、index. cshtml 和 about. cshtml。

【示例说明】

index. cshtml 中的@RenderPage("about. cshtml")将会在此位置显示 about. cshtml 页面中的内容。

6.2.5 Razor 中操作数据库

在 Web Matrix 3 中可以直接创建 SQL Server Express 2008 R2、MySQL 或 SQL

Server CE 数据库,也可以通过设置字符串连接的方式直接连接到已有的 SQL Server 或 MySQL 数据库中。

在这里,通过连接的方式连接第 4 章 SQL Server 的 HcoreManage 数据库。

单击 Web Matrix 3 中的数据库项,选择"新建"→SQL Server 菜单命令,弹出图 6-12 所示的参数设置对话框。填写正确后单击"确定"按钮。此时,在网站的数据库中出现 HscoreManage 数据库的名称,如图 6-13 所示。

Razor 中使用 database 类来操作数据库。表 6-2 是 Database 类的常用方法。

表 6-2　Database 类的方法

方 法 名 称	作　　用
Open()	使用连接名打开数据库连接
OpenConnectionString()	使用字符串打开数据库连接
Execute()	用于执行 Insert 或 Update
Query()	执行查询并返回集合
QuerySingle()	执行查询并返回一行一列的数据
QueryValue()	执行查询并返回一行一列的值
GetLastInsertId()	获取最后一条插入数据的 ID

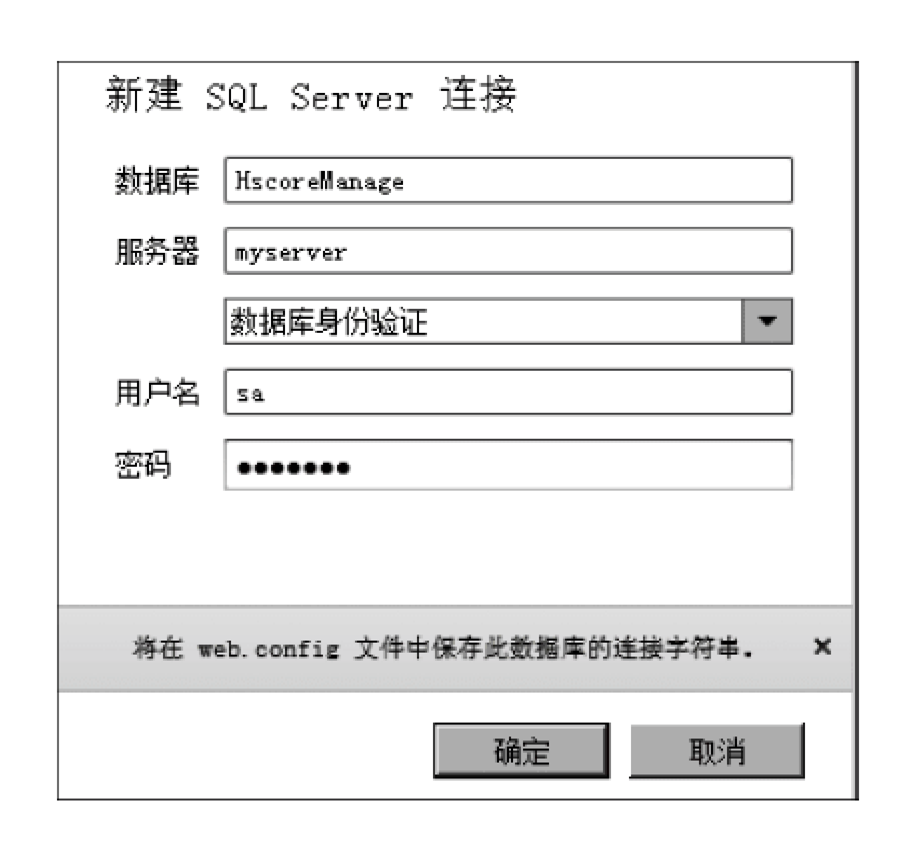


图 6-12　设置连接 SQL Server 参数



图 6-13　数据库连接设置成功

通常先使用 Database.open(数据库名称)打开指定的数据库,然后可以用 Query 方法执行 SELECT 语句返回集合,Razor 中返回的记录都是 dynamic 类型,也可以用 Execute 方法执行 UPDATE、DELETE 或 INSERT INTO 语句。

【例 6-13】 本例见资源包中的 6-13.cshtml。

【示例说明】

(1) 本示例的作用是在页面中以表格的形式显示 department 表中的记录。

(2) var db =Database.Open("HscoreManage")的作用是打开 HscoreManage 数据库。

(3) var list =db.Query("Select * from department")的作用是查询 department 表中的记录并返回 dynamic 集合。

(4) 通过 foreach(var item in list)语句遍历 dynamic 集合。

(5) @item. departno 表示当前 dynamic 集合中 departno 字段所对应的值。

在 Razor 中可以用 Webgrid 控件以表格的形式显示查询的结果。Razor 中的 Webgrid 类似于 Web Form 中的 GridView 控件。

【例 6-14】 本例见资源包中的 6-14. cshtml。

【示例说明】

(1) var grid=new WebGrid(list)语句是定义了一个名为 grid 的 webGrid 控件,其数据源(source 属性)是 list(查询 department 表中的记录并返回的 dynamic 集合)。

(2) @grid. GetHtml()是将 grid 以数据表的形式显示在网页中。

在例 6-14 中可以看出,默认情况下 webGrid 允许分页显示(每个页面显示 10 条记录)。它还有一些属性可以在实例化时进行设置,这些常用属性见表 6-3。

表 6-3 webGrid 控件实例化时的部分常用属性

属性及默认值	作　用
IEnumerable<dynamic> source	设置数据源
IEnumerable<string> columnNames = null	要显示的列名
string defaultSort = null	默认排序的字段
int rowsPerPage = 10	每页的行数
bool canPage = true	是否允许分页
bool canSort = true	是否允许排序

以上属性不能单独设置,需要在实例化时以命名参数的形式修改这些参数,如:

```
Var grid = new webGrid(source:list,rowsPerPage = 5)
```

这条语句定义了数据源为 list 的 webGrid 控件对象 grid,其分页时每页显示 5 条记录。

webGrid 控件的 GetHtml 方法也有一些属性,如表 6-4 所示。

表 6-4 webGrid 控件 GetHtml 方法的部分常用属性

属　性	作　用
string tableStyle = null	表格样式
string headerStyle = null	头部
string footerStyle = null	底部
string rowStyle = null	每一行的样式
string alternatingRowStyle = null	交替行
string selectedRowStyle = null	被选中
string caption = null	标题
bool displayHeader = true	是否显示头部
bool fillEmptyRows = false	是否填充空行
string emptyRowCellValue = null	空行里每个字段显示的值
IEnumerable<WebGridColumn> columns = null	在这里设置各列

续表

属　　性	作　　用
IEnumerable<string> exclusions = null	不显示的字段
string firstText = null	对应第一页链接显示的文字
string previousText = null	对应上一页链接显示的文字
string nextText = null	对应下一页链接显示的文字
string lastText = null	对应最后一页链接显示的文字
int numericLinksCount = 5	数字选择项的数目

上述属性也是在使用 GetHtml 方法时以命名参数的形式进行设置。

如@grid.GetHtml(tableStyle:"grid"),grid 的表格样式将会以设定的 grid 样式进行显示(grid 需要在样式中进行设置)。

【例 6-15】 本例见资源包中的 6-15.cshtml。

【示例说明】

本示例演示了上述部分属性的使用方法。

在 Razor 中可以使用基于 HTML 的标签、文本框、下拉列表框、单选按钮、命令按钮等客户端控件。这些控件是客户端的,因此在使用时不要加 Web Form 中的"runat=server"标记。

【例 6-16】 本例见资源包中的 6-16.cshtml。

【示例说明】

(1) 本示例演示了 Razor 在页面上使用基于 HTML 的标签、文本框、下拉列表框和命令按钮的方法。

(2) 本示例仅完成了向 class 表中添加记录的前台界面设计,没有进行后台的数据处理。

(3) 由于 class 表的 profid(专业编号)字段来自于 prof 表,因此,专业信息使用了下拉列表框,运行时将 prof 表的记录全部取出,使用 foreach 循环将专业信息以"专业名称(专业编号)"的形式添加到下拉列表框的选项中。

例 6-16 仅实现了添加班级信息的界面设计,要实现班级信息在数据库中的添加,需要借助 Database 类的 Execute 方法,即利用 Database 类的 Execute 方法执行 INSERT INTO 语句。

【例 6-17】 本例见资源包中的 6-17.cshtml。

(1) 本示例演示了 Razor 在页面上将记录插入数据表的方法。

(2) 注意 db.Execute 的使用。

6.3　Kooboo CMS 使用

Kooboo CMS 是一个比较灵活的 CMS,用它可以直接创建基于 C# 的 ASP.NET 网站。本节简要介绍 Kooboo CMS。

6.3.1 初识 Kooboo CMS

安装完 Kooboo CMS 后可直接在浏览器中输入 Kooboo CMS 的网站地址登录 Kooboo CMS。如果是在 Web Matrix 3 中使用 Kooboo CMS，可以在 Web Matrix 3 的 Kooboo CMS 网站中使用“在浏览器中浏览”，启动 Kooboo CMS。

Kooboo CMS 启动界面如图 6-14 所示，单击右上方的 LogOn，弹出图 6-15 所示的登录界面。Username 和 Password 的默认密码都是 admin，Redirect to 项改为 Admin page，然后单击 Login 按钮，进入 Kooboo CMS 的后台网站集群管理界面，如图 6-16 所示。



图 6-14 Kooboo CMS 在浏览器中的启动界面

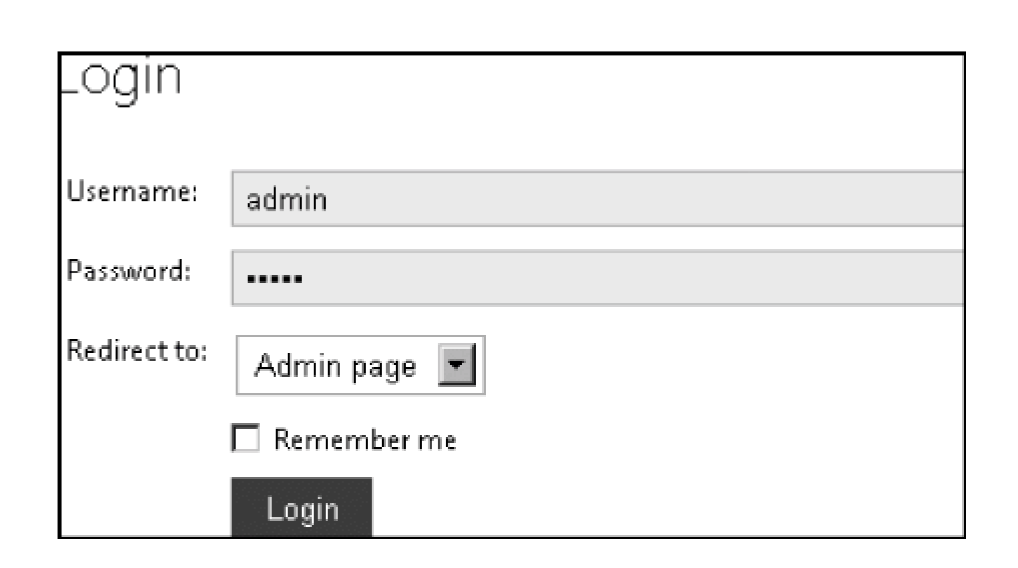




图 6-15 Kooboo CMS 登录界面



图 6-16 Kooboo CMS 网站集群

Site Cluster 代表一个网站集群，默认情况下有一个默认的示例网站 SimpleSite。可以将其他 Kooboo CMS 网站导入，也可以创建新的网站。

这里创建一个新的网站 demo，用来演示通过 Kooboo CMS 创建教师信息管理系统的方法。

如图 6-16 所示，单击“创建”图标按钮，选择 A new site，进入“新网站”的设置页面，如图 6-17 所示。

(1) Template：是已有的网站模板，Empty 是一个空网站，SampleSite 和 SmallBusiness 是两个不同风格的网站模板，这里选择 SampleSite 模板。

(2) 名称：是这个网站的存储名称，这里输入 demo。

(3) 显示名称：是在网页中显示的网站名称，这里输入 teacherInfo。

(4) 内容数据库：设置新建网站使用的数据库名称。这里可以选择其他网站建立的

数据库，也可以新建数据库。也就是说，Kooboo CMS 在多个网站中可以共享数据库。这里新建一个数据库 db。

（5）语言：Kooboo CMS 支持多语言。这里选择“中文(中华人民共和国)”。

（6）Time zone：时区。

单击“保存”图标按钮，即创建了名为 demo 的网站。



图 6-17　创建 Kooboo CMS 网站

6.3.2　创建 Kooboo CMS 网站开发步骤

创建 Kooboo CMS 网站的开发步骤如图 6-18 所示。

1. 建立数据库

Kooboo CMS 的数据库称为内容数据库，默认情况下是 XML 存储方式。XML 存储方式是最简单、最方便也是性能最差的一种存储方式。XML 存储方式没有任何额外数据结构需要创建，在开发过程中可以减少很多不必要的初始化和冲突的情况，减少开发成本。同时，XML 存储也是站点在导入、导出时任何的中间格式。

运行于任何数据库之上的站点，导出都是将内容存储在 XML 文件中；在另一个站点导入时，再将存储在 XML 的内容导入另一种数据库中。目前，Kooboo CMS 支持 XML、SQLCe、SQLServer、MySQL 和

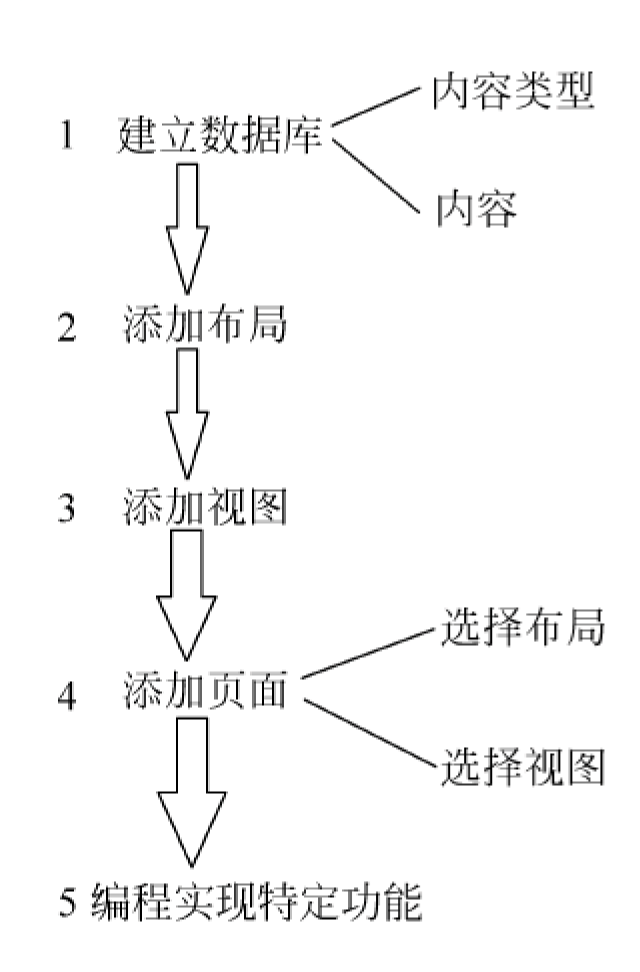


图 6-18　使用 Kooboo CMS 创建网站的步骤

MongoDB数据库。除了XML和SQLCe外，其他几个类型的数据库都需要进行配置。这里使用默认的XML数据存储方式。

内容数据库包括文本内容（TextContent）和媒体内容（MediaContent），文本内容的结构由文本内容结构类型（content type）负责定义，存储在文本内容目录中（text folder）。

因此，创建内容数据库分为两个步骤：第一步是先创建内容类型，内容类型就类似于数据库的表结构；第二步是创建内容，即text folder。

2. 添加布局

布局模板用于定义页面的布局结构。布局模板包含有HTML页面公共部分的代码，还预留有相应的占位符提供给不同的页面添加不同的内容。可以放到占位符的内容包括View、Html block、Module、Folder、Html content等。

Kooboo CMS提供了默认的布局模板，开发者可以利用默认的布局模板进行修改。

3. 添加视图

视图模板承担着组合HTML和内容数据的责任，开发人员查询数据，通过模板语法组合输出完整HTML内容。视图模板可以配置数据查询，配置的数据结果都会存在ViewData中，开发人可以通过ViewData["DataName"]或者ViewBag. DataName得到查询结果对象。视图模板中查询的数据，只允许本视图模板使用，不共享给其他视图。

视图模板可以有参数配置，这些参数可以在页面使用时设置参数值。在代码中使用参数值的方法是Page_Context. Current["parameter1"]。

4. 添加页面

页面承担着站点内容的组合、站点导航结构的构建。页面是由一系列配置信息组成，是Kooboo CMS站点处理请求的入口。在创建页面时，需要先选择布局模板，在页面设计器中可以设定页面需要的视图模板，也可以选择数据查询。布局模板和视图模式这两种模板加上数据查询就可以组成页面。

6.3.3 创建db内容数据库

db数据库包含两类信息：一类是部门信息（department），包括部门编号（departno）和部门名称（departname）；另一类是教师信息（teacher），包括教工号（tid）、姓名（tname）、出生年月（birth）、性别（sex）、职称（title）、所属部门（departno）。

下面就演示如何在Kooboo CMS中创建包含上述信息的内容数据库。

1. 创建department的内容类型

在demo网站管理界面中，选择“数据库”→“内容类型”，单击“创建”，如图6-19所示，在“创建内容类别”文本框中输入department，然后单击Create field按钮，弹出如图6-20所示界面。



图 6-19　创建内容类别

(1) 名称：即字段名。

(2) 标签：是字段的显示名。

(3) 控制类型：是在录入时对数据作校验的依据。

(4) 数据类型：相当于字段的数据类型。如果该字段的“概述字段”复选框被选中后，该字段会出现在内容项的中，“内容列表”复选框被选中后该字段才会出现在内容详细页中。

departno 的设置如图 6-20 所示。设置完成后，单击“保存”按钮即可。

基本　高级　验证

名称　departno
Name of your field, it uses the same naming rule as C# variable name.

标签　部门编号
Descriptive label of your field on the content editing page.

控制类型　Int32
The way that you would like to input your content.

数据类型　Int

默认值

☑ 概述字段
The Summary field will be used as a title or summary to describe your content item.

☑ 内容列表
Show this field in the CMS content list page.

保存　取消

图 6-20　创建 departno 字段

同理，departname 字段的设置如图 6-21 所示。

内容结构默认内置了一些系统字段(表 6-5)，但是在定义内容结构时，系统并不限制定义的字段名称。如果用户定义的字段名与系统字段相同时，则会自动将该字段的值存储在系统字段。这样的机制可以实现开发人员对系统字段的自由控制。

比如，系统内置 UserKey 字段，是让用户设置一个友好主键。但在默认的内容维护



图 6-21 创建 departname 字段

表单中，没有 UserKey 的输入框。此时，就可以在内容结构中添加一个 UserKey 字段，用于在内容维护表单中生成 UserKey 的输入框。

表 6-5 Kooboo CMS 的系统内置字段

字 段 名	类 型	描 述
Id	String	对关系型会存储自增长 ID
Repository	String	内容数据库名称
FolderName	String	目录名称
UserKey	String	内容友好主键
UtcCreationDate	DateTime	创建时间
UtcLastModificationDate	DateTime	最后修改时间
Published	Nullable<bool>	发布状态
SchemaName	String	内容结构名称(Content type)
ParentFolder	String	父内容的目录名称
ParentUUID	String	父内容的主键
UserId	String	编辑用户
OriginalFolder	String	内容广播时的源目录
OriginalUUID	String	内容广播时的源内容
IsLocalized	Nullable<bool>	是否本地化
Sequence	long	内容排序

2. 创建 teacher 的内容类型

teacher 的内容类型创建后如图 6-22 所示。teacher 中所属部门(departno)不需要创建，将 department 作为 teacher 的类别进行设置即可。

编辑: teacher

☐ Tree style data
Create the tree style data and management interface.

Create field

名称	标签	Control type	概述字段	内容列表	排序
tid	教工号	TextBox	YES	YES	1
tname	教师姓名	TextBox	YES	YES	2
sex	性别	TextBox	YES	YES	3
birth	出生日期	TextBox	YES	YES	4
title	职称	TextBox	YES	YES	5

图 6-22　teacher 的内容类型

3. 创建 department 的内容

选择“数据库→内容”，单击 New folder，弹出图 6-23 所示界面，设置见图上内容。

BASIC INFO | 权限设置 | 关联目录 | CONTE

名称　department

显示名称

内容类型　department

图 6-23　创建 department 内容

显示名称是在系统中该内容的显示文字，如果不设置，将会以名称项中的内容显示在系统中。

4. 创建 teacher 的内容

创建 teacher 的内容如图 6-24 所示。

BASIC INFO | 权限设置 | 关联目录 | CONTE

名称　teacher

显示名称

内容类型　teacher

图 6-24　创建 teacher 内容

在这里，需要将 department 目录设定 teacher 的类别目录。单击“关联目录”选项卡，如图 6-25 所示，在“类别目录”下拉列表框中选择 department，选中“单选”复选框，保存。

本示例不用设置内嵌目录。内嵌目录本质上是一对多的数据关系，如论坛中某条留言和留言的回复，回复的内容就可以作为留言的内嵌目录。

5. 添加数据

单击 department 内容，弹出页面如图 6-26 所示，单击 Add content 图标按钮，弹出页面如图 6-27 所示，这里输入的就相当于表中的记录。选中 Published 复选框，该条记录才能在页面中看到。db 数据库中添加的部门表信息如图 6-28 所示。



图 6-25　将 department 目录作为 teacher 的类别目录



图 6-26　department 的 add content（1）

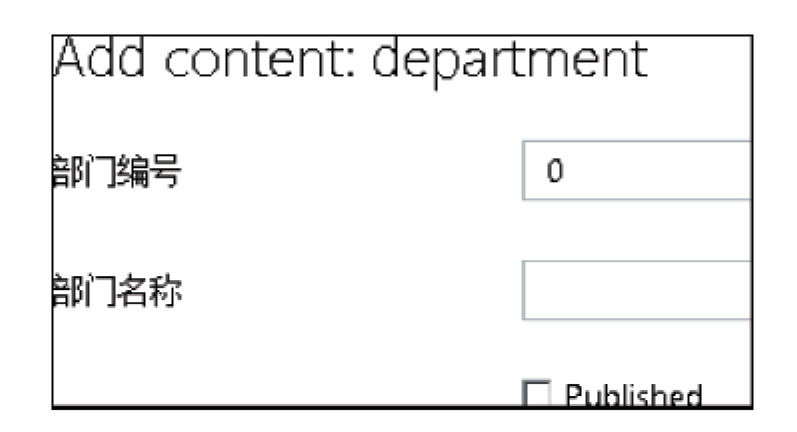


图 6-27　department 的 add content（2）

department

部门编号	部门名称	Creation date	Published
6	影视管理	2015/5/27	YES
5	工商管理	2015/5/13	YES
4	金融	2015/5/13	YES
3	会计	2015/5/13	YES
2	信息管理	2015/5/13	YES
1	电子商务	2015/5/13	YES

图 6-28　department 手工添加的内容

添加“教师表”的页面如图 6-29 所示，由于 teacher 以 department 目录为类别，所以在这里需要选择 department 中的相关信息。db 中添加的 teacher 信息如图 6-30 所示。



图 6-29　为 teacher 手动添加内容

teacher

教工号	教师姓名	性别	出生日期	职称	Creation date	Published
js006	刘老师	女	1962-8-9	教授	2015/5/28	YES
js008	test	男	1969/02/01	高工	2015/5/18	YES
js007	陈老师	女	1969-3-4	高级工程师	2015/5/13	YES
js005	郭老师	女	1988-8-26	讲师	2015/5/13	YES
js004	王老师	男	1987-5-1	讲师	2015/5/13	YES
js003	彭老师	男	1977-12-5	副教授	2015/5/13	YES
js002	李老师	男	1985-3-2	讲师	2015/5/13	YES
js001	张老师	男	1975-3-2	副教授	2015/5/13	YES

图 6-30　teacher 内容中手动添加的所有记录

6.3.4　创建布局

布局用来统一网站的风格。选择“开发”→“布局”→“创建”菜单命令，进入布局的设计界面。如图 6-31 所示。可在此直接输入布局的代码，也可以利用布局右侧的辅助功能使用已有的布局模板。



图 6-31　布局设计界面

在 New layout 文本框中输入布局的名称 onecolumn，然后单击右侧“布局帮助”中的“布局”，展开布局选项，单击 OneColumn 名称布局右侧的按钮，如图 6-32 所示，系统弹出对话框，询问是否确定替换模板，单击“确定”按钮，系统会自动创建 onecolumn 布局。



图 6-32　使用布局帮助生成布局代码

@Html. FrontHtml(). Position("main")中的"@Html. FrontHtml(). Position()"是 onecolumn 布局中的占位符代码，每个 Position 都可以使用一个视图。在布局模板中也可以对数据进行操作。

【例 6-18】 onecolumn 布局模板代码请见资源包中的 6-18. txt。

在名称为 header 的 Position 中，将用来显示网站的导航菜单以及进行用户登录的操作。

6.3.5 创建视图

1. 视图名称

选择“开发”→“视图”→“创建”菜单命令，进入视图的编辑模式，如图 6-33 所示。

kooboo 保存 后退 Site ter

New view Menu

```
<nav class="nav">
 <ul>
 @foreach (var item in MenuHelper.Top())
 {
     if (item.Navigation.Show)
     {
         <li class ='@(MenuHelper.IsCurrent(item) ? "active" : "")'>@Html.FrontHtml().PageLink(item.LinkText, item.FullName)</li>
     }
 }</ul>
</nav>
```

图 6-33 创建视图

视图用于处理数据，图 6-33 所示为一个用于显示菜单的名为 Menu 的视图。

如果需要按照类别存放视图，可采用“类别名. 视图名”的方式建立视图，这样，Kooboo CMS 会在视图中先建立一个类别名，之下才是视图名。

如图 6-34 所示，创建一个 depart. add 的视图(本视图用于添加部门信息)。保存此视图后，视图菜单中显示的是一个名为 depart 的类目，展开该类目后，才能在右侧看到 depart. add 视图，如图 6-35 所示。

kooboo 保存 后退 Site te

New view depart.add

```
<div class="common-form">
   <h5 class="title">添加部门信息</h5>
   <form method="post" action="@SubmissionHelper.CreateContentUrl()" >
     @Html.AntiForgeryToken()
    <input type="hidden" name="FolderName" value='@SecurityHelper.Encrypt("department")'  />
    <input type="hidden" name="Published" value="true" />
     <input type="hidden" name="RedirectUrl" value='@Url.FrontUrl().PageUrl("departs")'/>
     <p class="field">
    <label for="departno1">部门编号:</label>
     <input type="text" id="departno" name="departno"  data-val-required='@("departno is required".Label())' data-val="true" />
      @Html.ValidationMessageForInput("departno")
    </p>
    <p class="field">
    <label for="departname1">部门名称:</label>
    <input type="text" id="departname" name="departname"  data-val-required='@("departname is required".Label())' data-val="true"
/>
    @Html.ValidationMessageForInput("departname")
    </p>
    <p class="buttons">
   <button type="submit">保存</button>
  </form>
</div>
```

图 6-34 视图名称

2. 创建数据查询

在视图中,可以使用数据查询来打开数据集。比如要创建一个用于浏览部门信息的视图 depart. info,可先创建一个部门信息的数据查询。如图 6-35 所示,单击数据查询的"+"按钮,在弹出的 Edit Filters 页面中选择 department,如图 6-36 所示,单击"下一条"按钮,如图 6-37 所示,设置 Data Name(本例设为 dsdepart)。

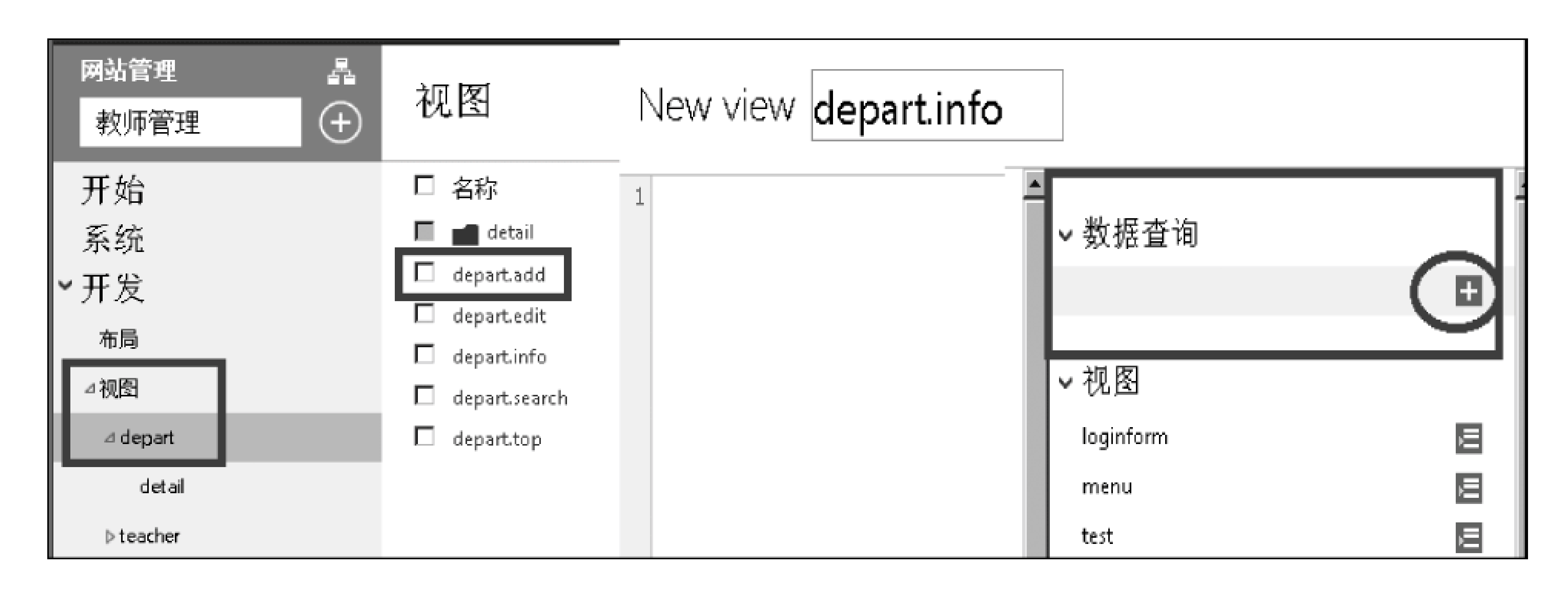


图 6-35 视图中创建数据查询(1)



图 6-36 视图中创建数据查询(2)

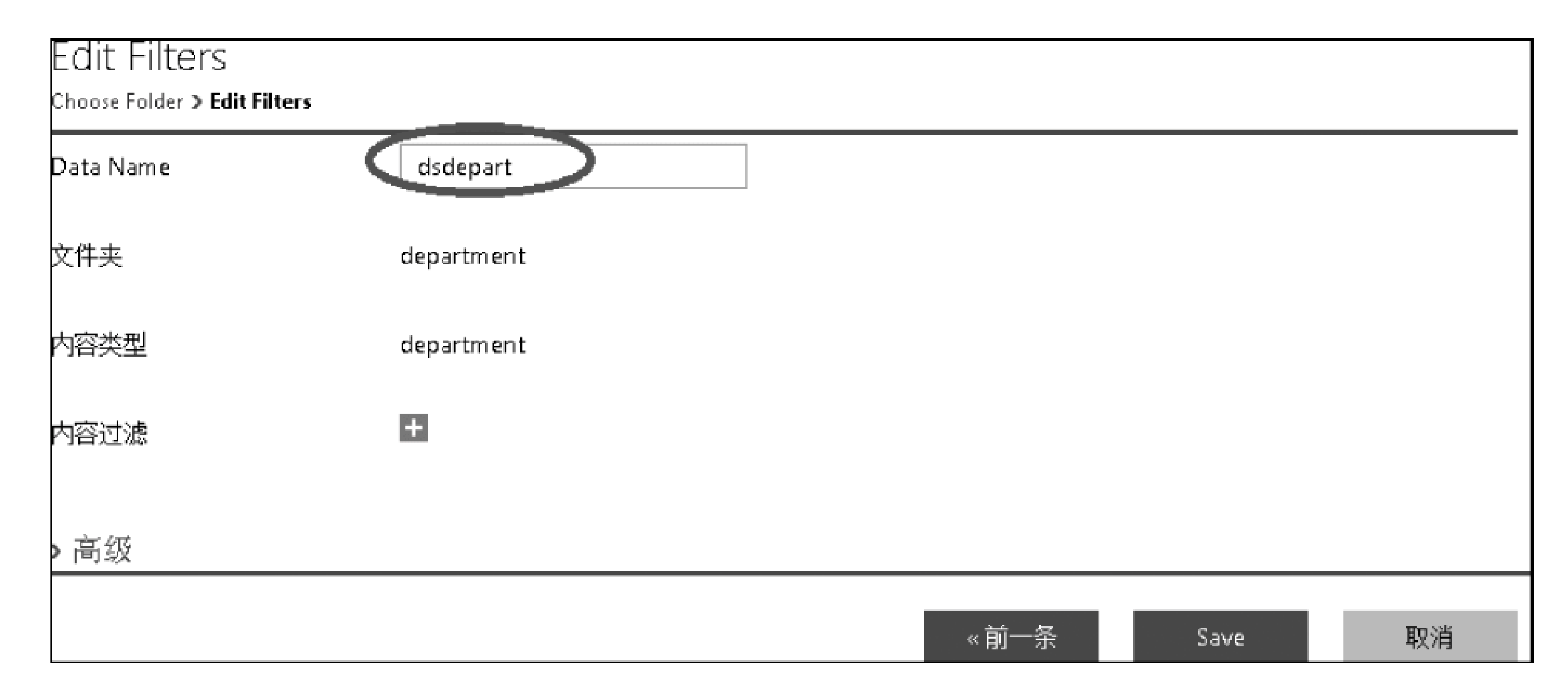


图 6-37 视图中创建数据查询(3)

如果要设置筛选规则，可利用“内容过滤”来设置。要对数据查询做更多的设置，可单击“高级”选项，如图 6-38 所示。

高级
TOP
Sort filed　departno
排序方向　ASC
启用分页　☑
每页记录数　5
A const value OR dynamic value get from query string. eg: 10 OR {PageSize}
分页下标　{PageIndex}
The page index parameter name. eg:{PageIndex}
«前一条　Save　取消

图 6-38　视图中创建数据查询(4)

(1) Sort filed：用来选择排序的字段名，数据查询将按照此字段名进行排序。

(2) 排序方向：ASC 为递增，DESC 为递减。

(3) 启用分页：默认为不分页。如启用分页，则数据查询将按照每页记录数中设置的记录数量进行分页显示。

(4) 每页记录数：启用分页时才起作用，是分页时每页中显示的记录数量。

(5) 分页下标：用于控制分页时的页码。

如果启用分页，则要在视图中添加分页的操作代码。depart.info 视图启用分页，每页显示 5 条记录。单击 Save 按钮后，数据查询就创建成功了。

3. 使用数据查询

如果在视图中创建了数据查询，在视图中使用“ViewBag.数据查询名称”就可以访问数据查询中的数据了。depart.info 视图代码如图 6-39 所示，该视图以表格的方式浏览 department 表中的内容，并以每页 5 条记录的方式进行分页。

(1) @Html.FrontHtml().PageLink(item.departno,"departs/departedit", new {UserKey= item.UserKey })这行代码的作用是将 item.departno 设为超链接，当单击显示出来的某一条记录的 departno 时，页面将会被导航到 departs/departedit 页面。Kooboo CMS 是以匿名方式传递页面参数，new {UserKey=item.UserKey }就是传递的页面参数，这行代码的作用相当于 Web Forms 中的 Response.Redirect("departs/departedit UserKey=×××")。UserKey 是对应记录的 UserKey，departedit 是记录详细编辑页，导航将显示该条记录所对应的详细页。

(2) @{}里面的是分页的设置。

4. 数据的操作

对数据表的增、删、改、查也是在视图中完成的。Kooboo CMS 提供了以下数据操作

New view depart.info

```
@using Kooboo.CMS.Content.Models
<div>
   <table width="50%" border="1">
     <caption>部门信息</caption>
     <tr align="center">
      <th  width="30%" >部门编号</th>
       <th width="40%" >部门名称</th>
     </tr>
     @foreach(var item in ViewBag.dsdepart)
     {
       <tr align="center">
       <td>
         @Html.FrontHtml().PageLink(item.departno,"departs/departedit", new { UserKey= item.UserKey })</td>
       <td>@item.departname</td>
     </tr>
     }
   </table>
</div>
@{
    var pageOption = new PagerOptions
    {
        NextPageText = "Next »",
        PrevPageText = "« Previous",
        CssClass="pagination",
        CurrentPagerItemWrapperFormatString="<span class=\"current\">{0}</span>",
        NavigationPagerItemWrapperFormatString="<span class=\"pagenavigation\">{0}</span>"
    };
}
@Html.FrontHtml().Pager(ViewBag.dsdepart,pageOption)
```

图 6-39　在视图中使用数据查询

方法。

@SubmissionHelper. DeleteContentUrl(string FolderName,string uuid)：删除指定 Folder 的 uuid 所对应的记录。

@SubmissionHelper. UpdateContentUrl()：修改记录。

@SubmissionHelper. CreateContentUrl()：添加记录。

1）删除记录

通常是在浏览记录的时候为每条记录设置一个 form,每个 form 对应一个用于删除的按钮(input type="submit"),删除的 FolderName 和 uuid 参数则使用隐藏的表单标记以 post 方式传递。下面是 teacher 表进行浏览和删除的视图示例。

```
@foreach(dynamic item in ViewBag.dstea)
{
    < form method = "post" action = '@SubmissionHelper.DeleteContentUrl(
    @Request["FolderName"],@Request["uuid"])'>
    @Html.AntiForgeryToken()
    < tr align = "center" >
    < td >< a href = '@Url.FrontUrl().PageUrl("teachers/teacheredit", new { UserKey =
    item.UserKey})'@ViewHelper.Edit(item,"tid") >@item.tid</td>
    < td>@item.tname</td>
    < td>@item.sex</td>
    < td>@item.title</td>
    < td>@item.birth</td>
    < td>
    @foreach(var category in ((Kooboo.CMS.Content.Models.TextContent)
    item).Categories(ContentHelper.TextFolder("department")))
    {
```

```
            @category["departname"]
        }
        </td>
        <td>
        <input type="submit" name="submit" onClick="javascript:affirm();" value="删除" />
        <input type="hidden" name="FolderName" value=
        '@SecurityHelper.Encrypt("teacher")'/>
        <input type="hidden" name="uuid" value="@item.UUID" /></td>
        </tr>
        </form>
}
```

其中,下面的部分是以隐藏的表单标记传递删除方法的两个参数。

```
<input type="hidden" name="FolderName" value=
'@SecurityHelper.Encrypt("teacher")'/>
<input type="hidden" name="uuid" value="@item.UUID" />
```

【小提示】

表单中的控件标记当以 post 方式传递时,需要设置 name 属性,即将 name 设置为 DeleteContentUrl 方法所需要的参数名称,然后在 form 的 action 方法中采用 Request["参数名"]的方式即可获取到。

由于 teacher 表中使用了 department 表作为类别,下面是读取 teacher 表中某条记录的类别所用的代码。

```
@foreach(var category in ((Kooboo.CMS.Content.Models.TextContent)
item).Categories(ContentHelper.TextFolder("department"))) {
    @category["departname"]
}
```

2) 添加记录

下面是添加教师信息的部分代码。

```
<form id=" form" method="post" action="@SubmissionHelper.CreateContentUrl()">
@Html.AntiForgeryToken()
<input type="hidden" name="FolderName" value=
'@SecurityHelper.Encrypt("teacher")'  />
<input type="hidden" name="Published" value="true" />
<input type="hidden" name="Categories[0].FolderName" value="department" />
    ⋮
<p class="field">
<label for="depart1">所属部门:</label>
<select id="Categories[0].UUID"   name="Categories[0].UUID"   >
@foreach(var myitem in ViewBag.depart) {
    <option value  = '@myitem.UUID'>@myitem.departname</option>
```

```
}
</select>
</p>
<p class="buttons">
<button type="submit">保存</button>
</form>
```

添加的方式与删除的方式类似，即将 UpdateContentUrl 所需要的各个参数值以指定 name 的表单标记 post 过去即可。Kooboo CMS 会自动接收，然后进行添加操作。对于不需要用户看到的表单标记，采用隐藏方式(input type="hidden")进行设置。

3) 修改记录

修改与添加的方式很接近，区别在于修改需要将对应记录的值显示在对应的表单标记中，即让用户看到该条记录的值。可使用带内容过滤的数据查询来得到指定的记录，然后进行显示。

下面是 teacher 表修改记录的视图的部分代码。

```
<form name="messageForm" method="post" action=
"@SubmissionHelper.UpdateContentUrl()" >
@Html.AntiForgeryToken()
<input type="hidden" name="FolderName" value=
'@SecurityHelper.Encrypt("teacher")'  />
   ⋮
@foreach(var item in ViewBag.tea)  {
    <input type="hidden" name="uuid" value="@item.UUID" />
           ⋮
}
<p class="buttons">
<button type="submit">保存</button>
</form>
```

本例中的数据查询 tea 创建方法如图 6-40 所示。内容过滤需要指定 UserKey=={UserKey}，这样，当使用 @Html.FrontHtml().PageLink(item.tid," teachers/teacheredit", new { UserKey= item.UserKey })导航到此页时，tea 查询就会根据接收到的 UserKey 值得到指定的记录了。

4) 查询记录

下面的代码是执行按所属部门查找教师的查询功能的核心代码。

```
@functions{
    TextFolder newsFolder= ContentHelper.TextFolder("teacher");
    TextFolder categoryFolder  =  ContentHelper.TextFolder("department");
    public IEnumerable<TextContent> GetByCategory(string uuid) {
        return newsFolder.CreateQuery().WhereCategory(categoryFolder.
        CreateQuery().WhereEquals("uuid", uuid));
    }
```

图 6-40　设定数据查询的内容过滤

```
 }
@helper setds(string s) {
      ViewBag.dstea = GetByCategory(s);
}
<form method = "post" action = '@setds(@Request["bydepart"])'>
@("按部门查找教师".Label())
<select id = "bydepart" name = "bydepart">
<option value = '0'>请选择部门</option>
@foreach(var de in ViewBag.depart){
     <option value = '@de.UUID'>@de.departname</option>
}
</select>
<input type = "submit" id = "submit" name = "submit"  value = "查找" />
</form>
```

setds(string s)的作用是用于建立一个按选定的部门查找教师的数据查询。数据查询的来源则是利用 GetByCategory(string uuid)函数得到。

newsFolder.CreateQuery().WhereCategory(categoryFolder.CreateQuery().WhereEquals("uuid", uuid))是 Kooboo CMS 提供的进行数据查询的方法。

下述代码见资源包。

【例 6-19】　Menu 视图代码。

【例 6-20】　LoginForm 视图代码。

【例 6-21】　teacher.top 视图代码。

【例 6-22】　teacher.info 视图代码。

【例 6-23】　teacher.edit 视图代码。

【例 6-24】　teacher.add 视图代码。

【例 6-25】　teacher.search 视图代码。

【例 6-26】 teacher. searchbydepart 视图代码。
【例 6-27】 teacher. detail. return 视图代码。
【例 6-28】 depart. top 视图代码。
【例 6-29】 depart. info 视图代码。
【例 6-30】 depart. edit 视图代码。
【例 6-31】 depart. add 视图代码。
【例 6-32】 depart. search 视图代码。
【例 6-33】 depart. detail. return 视图代码。

6.3.6 创建页面

选择“网站管理”→“页面”→“新建”，如图 6-41 所示，需要先选定页面的布局名称。这里选择之前创建的 onecolumn 视图，进入页面编辑器，如图 6-42 所示。



图 6-41 创建页面



图 6-42 页面编辑器

在页面设计器标签中的 New page 文本框中输入页面的名称(本例是 teachers)，Place Holer 的位置是占位符，可以添加视图、目录或标签。这里添加两个视图：teachers. top 和 teachers. info。

在“设置”选项卡中，将 teachers 页设置为首页，如图 6-43 所示，即网站一进入就显示的页面。设置为首页的页面一个网站中只能有一个。

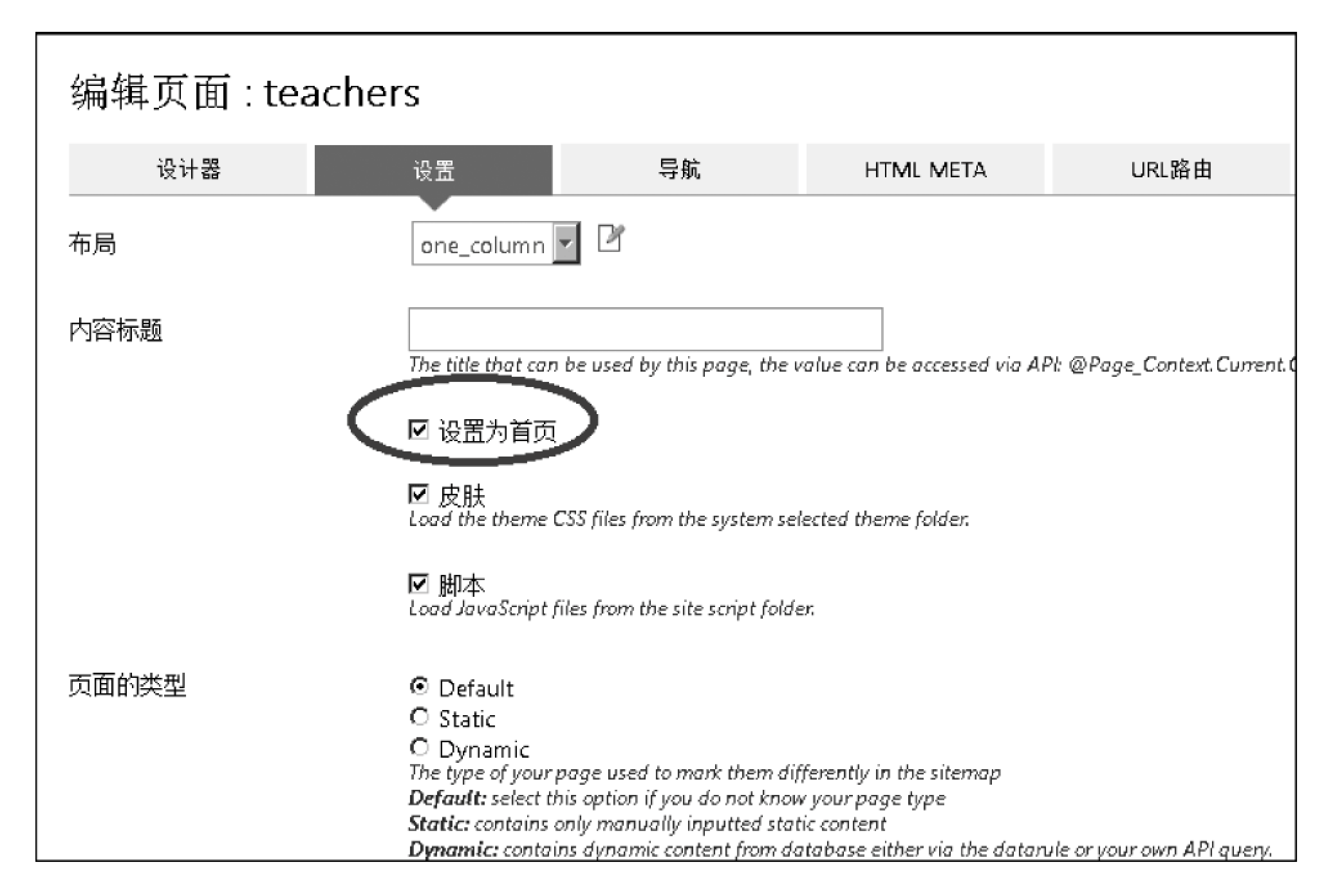


图 6-43　页面中的“设置”标签

在“导航”选项卡中，如图 6-44 所示。如果将页面显示在导航，则可以通过@MenuHelper 命令在视图中进行页面导航。

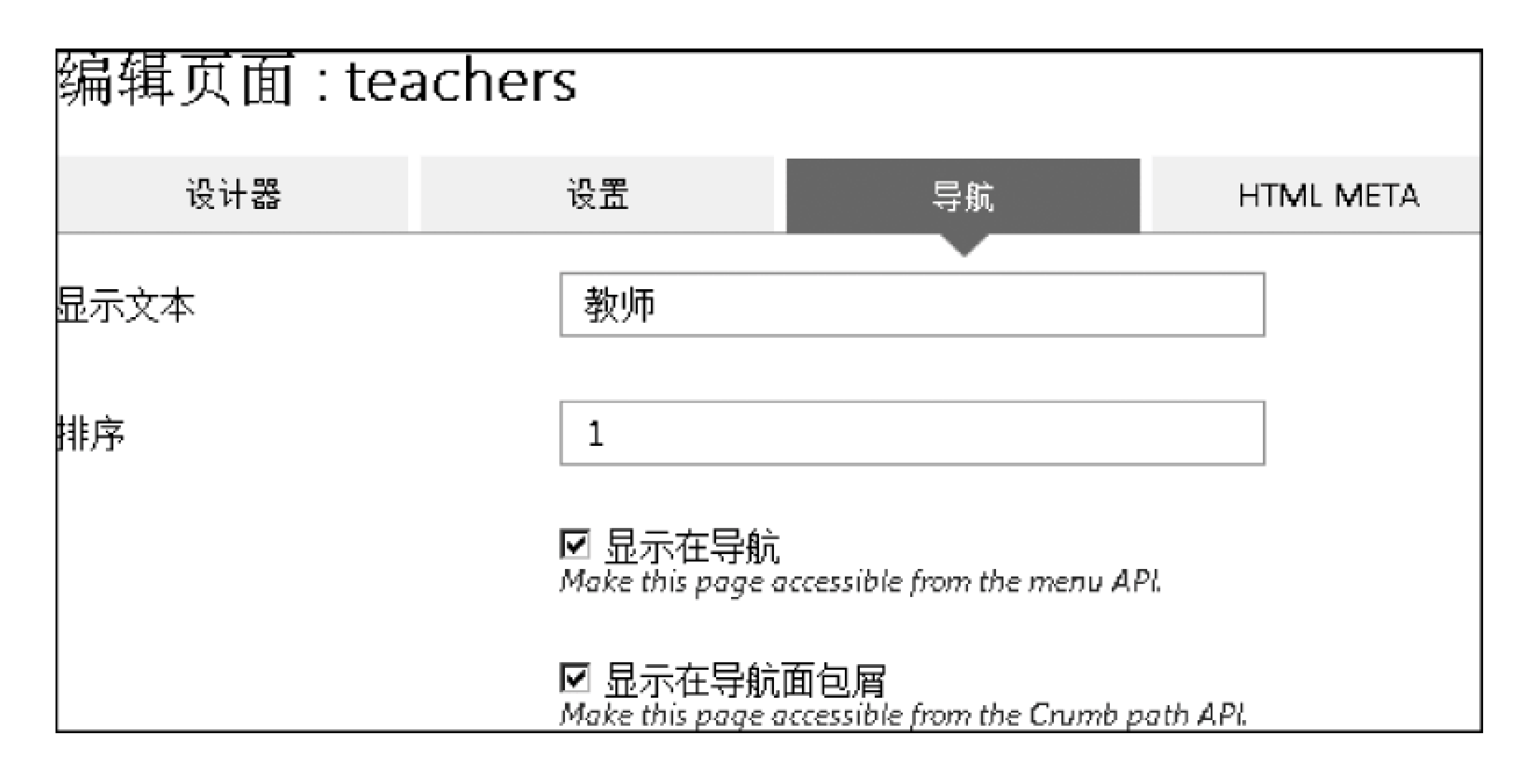


图 6-44　页面中的导航设置

(1) 显示文本是该页面显示在导航栏里的文本。

(2) 排序中的序号用于设置页面在导航中的顺序。

在“URL 路由”选项卡中，如图 6-45 所示，可以设置以下内容。

(1) Link target：页面显示的位置，有_self、_blank、_parent 和_top 这 4 种方式。



图 6-45　页面中的“URL 路由”选项卡设置

（2）External URL：设置页面的外部访问路径。

（3）URL 别名：用于设置页面访问时的备注名。

（4）URL 路径：由于本示例需要用 new { UserKey=item. UserKey}传递匿名参数，此处设置为{UserKey}，其作用相当于指明“teachers UserKey=×××”页面的 UserKey 参数。

（5）默认值：本示例添加 UserKey（注意不要用括号），这个 UserKey 就相当于“teachers UserKey=×××”的×××。

如果要创建类似于 teachers/teacheredit 中的 teacheredit 子页面，则可以在“开始”中的 teachers 页面添加 New sub page 子页面，如图 6-46 所示，选择布局后，其他的操作就和上面页面设置类似了。

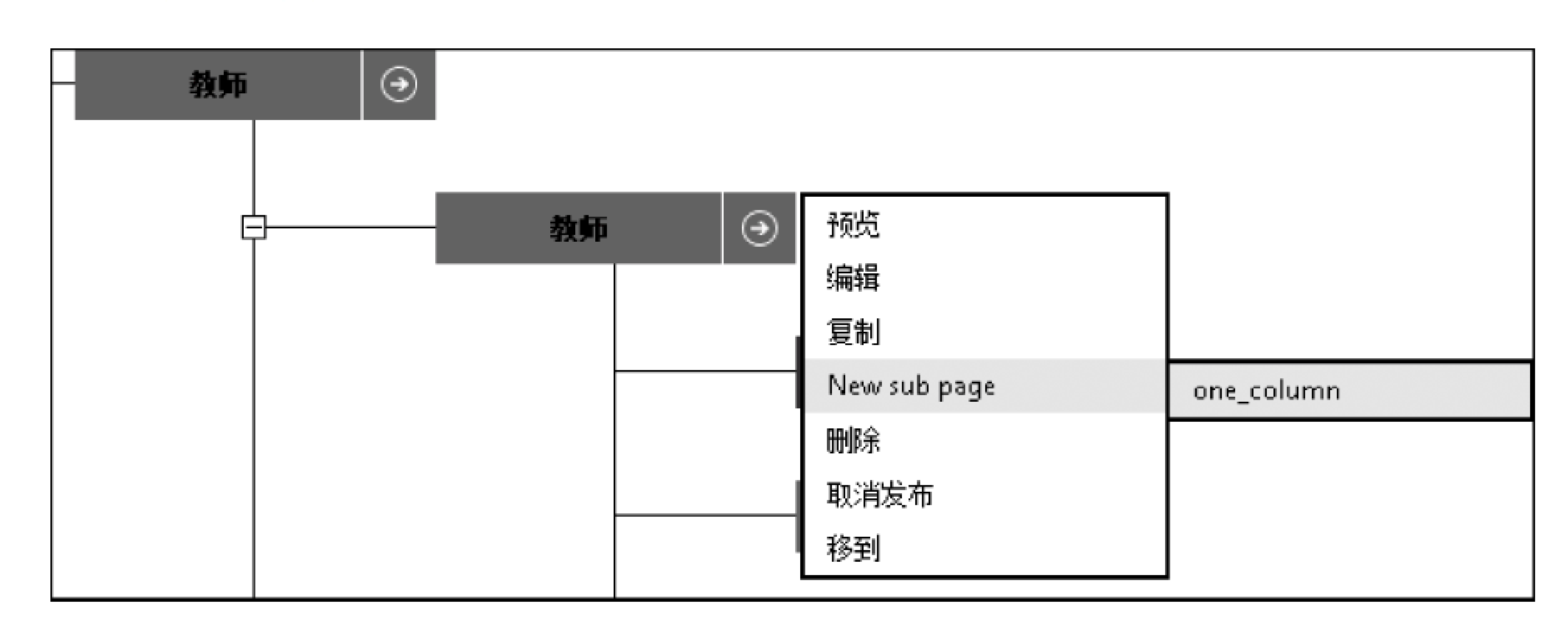


图 6-46　添加 New sub page 子页面

本系统最终的页面结构如图 6-47 所示。

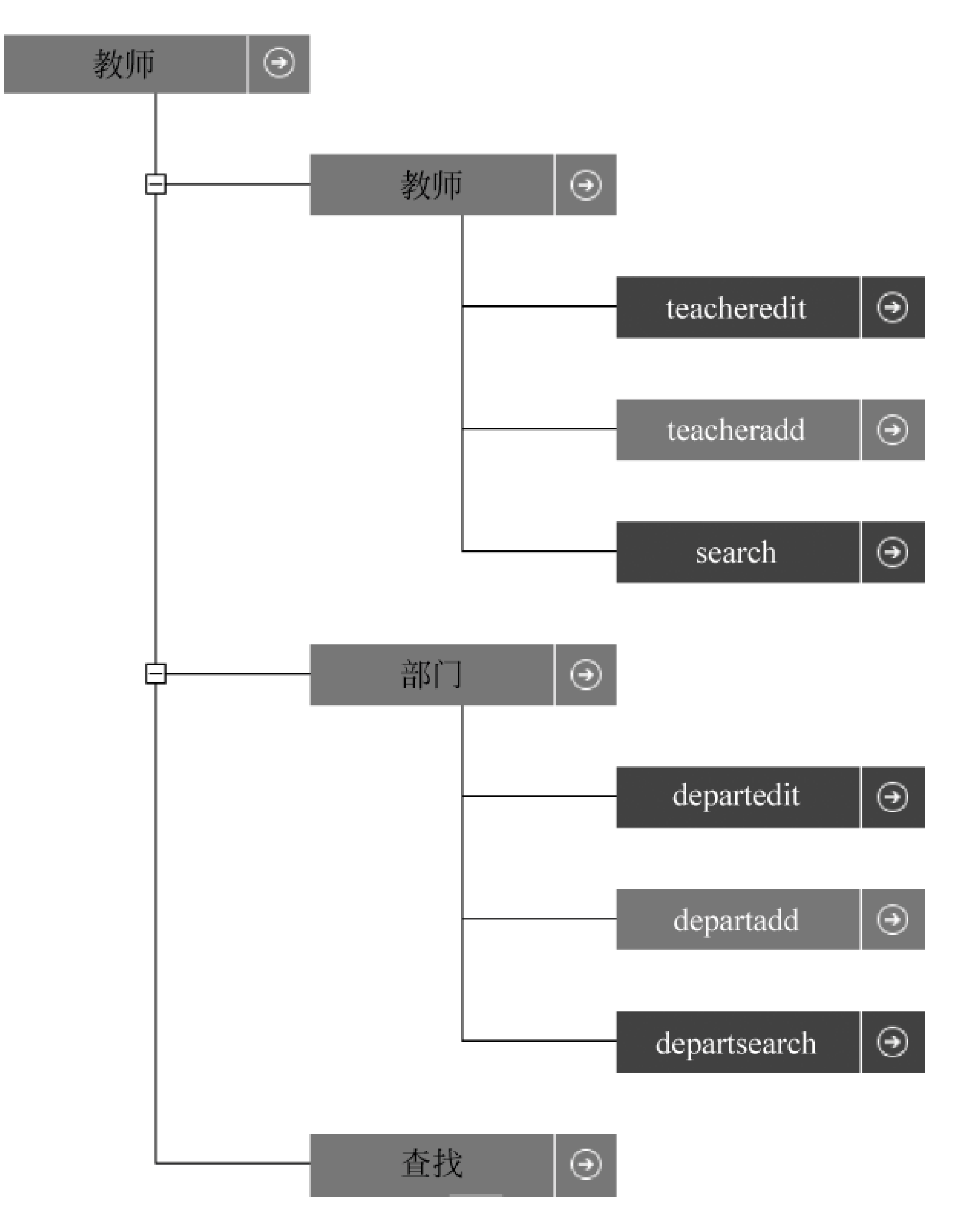


图 6-47 demo 中的页面结构



本章小结

本章讲述了利用 CMS 建站的方法。本章分为三部分：第一部分介绍 CMS 的相关概念；第二部分介绍了 Web Matrix 的使用和 Razor 语法；第三部分是本章的重点，通过建立一个教师管理的网站介绍了 Kooboo CMS 系统的使用。



思考与练习

1. 什么是 CMS? 它有哪些特点?
2. 有哪些比较典型的 CMS 软件?
3. Web Matrix 3.0 如何安装和使用?
4. Razor 语法如何使用? 如何使用 Razor 语法进行数据库操作?
5. 简述 Kooboo CMS 创建网站的流程。

第 7 章

网站建设实例

为了使读者能够真实地感受网站建设过程，结合网站开发具体要求和软件工程思想，本章通过介绍一个具体网站开发案例来综合课程知识点，阐述网站开发的真实过程和方法，详细描述了一个企业网站开发项目实施的具体步骤。

7.1 前期设计

需求分析、总体设计以及数据库设计都是软件工程中必不可少的环节，这些在开发之前的设计称为前期设计，网站的开发也需要这些内容。

7.1.1 需求分析

在网站开发之前所做的设计方案往往会对网站的最终效果产生巨大的影响，许多问题在开发之前就应该进行深入的调查和讨论。例如，这个网站面向的用户是谁？用户所需要的主要功能有哪些？日常处理的数据集有多大？是否需要使用数据库服务器？用户所使用的主要操作系统有哪些？

通过对用户初始的调查，系统设计者可以了解用户的现有状况和对最终系统的需求，在此基础上，进行可行性分析，也就是具体分析系统的开发是否存在内外部的必要条件，包括技术上、资金上、经济上、人力资源以及组织管理上的必要条件，同时还要分析在目前条件下是否有必要开发这样一个信息管理系统。

在可行性分析完成之后，就应该进行具体的需求分析、完成数据库的设计以及确定网站的主要功能。在网站建设之前，需要认真做好规划。在规划阶段，应该让用户更多地参与进来。无论多么详细的规划，在网站实施过程中也需要不断地进行修改，并且不断地进行讨论，认真听取用户的反馈意见，以最终确定网站的建设方案。

7.1.2 网站总体设计

本章所开发的简单网上书店充分利用了网络的优势，使管理者能够方便、快捷地进行

用户管理、图书管理、库存管理和购物车等功能。主要页面与功能如下。

1. 首页

首页主要提供了简单网上书店系统的导航功能,包括用户登录、用户注册、图书查询、新书入库、封面上传、库存管理、购物车等功能。

2. 用户登录功能

输入正确的用户名和密码,登录成功,进入首页;否则出现提示信息。

3. 用户注册功能

通过用户注册,可以添加新的用户信息。注册成功的用户可以通过登录进入系统。

4. 图书管理功能

图书管理主要包括新书入库、封面上传、图书查询、库存管理、查看图书介绍等功能。

5. 购物车功能

购物车主要包括购物车的查看、购书数量的修改、结账、清空购物车等功能。

7.1.3 数据库设计

数据库是系统数据层的实现,在前面详细分析了该网站所要完成功能的基础上,进行数据分析,从而设计了本网站的数据库模型。

数据库 MyBook 包括以下 5 个表:图书信息表 books、用户信息表 users、购物车表 shoppingCart、客户订单状态表 orders、客户订单详细信息表 orderDetails。表结构如表 7-1～表 7-5 所列。

表 7-1 图书信息表 books

字段名称	数据类型	长度	是否允许空值	说 明
book_id	varchar	6	否	图书编号,设为主键
title	varchar	80	否	书名
type	varchar	20	否	图书类型
price	decimal	5	否	单价
notes	varchar	250	否	图书简介
inventory	int	4	是	库存量

表 7-2 用户信息表 users

字段名称	数据类型	长度	是否允许空值	说 明
user_id	varchar	50	否	用户编号,设为主键,为自动编号
user_name	varchar	20	否	用户名
user_pwd	varchar	20	否	用户密码
user_email	varchar	80	是	用户电子邮箱

表 7-3　购物车表 shoppingCart

字段名称	数据类型	长度	是否允许空值	说　明
user_id	varchar	50	否	用户编号
book_id	varchar	6	否	图书编号
title	varchar	80	否	书名
quantity	int	4	否	购书数量
price	decimal	5	否	图书价格

表 7-4　客户订单状态表 orders

字段名称	数据类型	长度	是否允许空值	说　明
order_id	int	4	否	订单编号
user_id	varchar	50	是	用户编号
status	varchar	20	是	订单状态

表 7-5　客户订单详细信息表 orderDetails

字段名称	数据类型	长度	是否允许空值	说　明
order_id	int	4	否	用户编号
book_id	varchar	6	是	图书编号
title	varchar	80	是	书名
quantity	int	4	是	订购数量
price	decimal	5	是	图书单价

7.2　网站详细设计与开发

简单网上书店是在 Visual Sudio 2013 的环境里开发的，首先需要在 Visual Sudio 2013 中新建网站，并定义网站的名称和路径。当新网站创建完成后，Visual Sudio 2013 就会创建一个新的 Web 应用程序。

在新的 Web 应用程序中，Visual Sudio 2013 将为网站添加一个新页面，并在编辑器中将它打开。这个新的 ASP.NET 页面需要先进行界面设计，在页面上添加合适的控件。当页面上所有控件都定义完成后，需要保存和浏览页面。此时的 ASP.NET 页面只是一个界面，不具有处理事件的实际能力，需要添加必要的代码，才能完成相应的功能。

代码添加完成后，保存 Web 窗体页面并生成了应用程序，单击工具栏中的“运行”按钮，程序自动编译，最后生成结果页面，这时，就能在一个内嵌的浏览器窗口中查看页面。

7.2.1　Web. config 文件

Web. config 文件是一个基于 XML、人机可读的文件，包含应用程序的配置选项。ASP.NET 的配置文件是基于 XML 格式的纯文本文件，存在于应用的各个目录下，统一命名为 Web. config。它决定了所在目录及其子目录的配置信息，并且子目录下的配置信息覆盖其父目录的配置。这种方式可以为应用程序建立一套有层次的配置机制。如果在

应用程序的主目录下添加了 Web.config 文件，其中包含的配置设置在整个应用程序中有效。

本例中 Web.config 中需要设置的代码有两部分：一是设置数据库连接；二是设置安全身份验证模式为 Forms 模式。代码及说明如下。

（1）设置数据库连接。

```
<appSettings>
<add key="connstring"
value=" DataSource=(local);Initial Catalog=mybook; Integrated Security=True;"/>
</appSettings>
```

【小提示】

以上代码为 Windows 身份验证模式，数据库名为 mybook。

```
<appSettings>
<add key="connstring"
value=" server=(local);Database =mybook; Uid=sa;Pwd=d"/>
</appSettings>
```

【小提示】

以上代码为 SQL Server 身份验证模式，数据库名为 mybook，用户名为 sa，密码为 d。

（2）设置安全身份验证模式为 Forms 模式。

```
<authentication mode="Forms">
<forms name=".ASPXAUTH" loginUrl="login.aspx" timeout="40"></forms>
</authentication>
```

【语法说明】

通过＜authentication＞节可以配置 ASP.NET 用来识别进入用户的安全身份验证模式。可能的模式是 Windows、Forms、Passport 和 None。

None：不执行身份验证。

Windows：IIS 根据应用程序的设置执行身份验证（基本、简要或集成 Windows）。在 IIS 中必须禁用匿名访问。

Forms：为用户提供一个输入凭据的自定义窗体（Web 页），然后在应用程序中验证他们的身份。用户凭据标记存储在 Cookie 中。

Passport：通过 Microsoft 的集中身份验证服务执行，它为成员站点提供单独登录和核心配置文件服务。



【小提示】

以上代码为设置 Form 验证，禁止用户从网站的非登录页面登录网站；Cookie 对象名为.ASPXAUTH；未通过验证或超时后重定位的页面为登录页面 login.aspx；Cookie 失效时间为 40min。

7.2.2　系统功能划分

根据简单网上书店的基本要求，将整个网站功能划分为用户登录、用户注册、按名搜索、查看图书介绍、放入购物车、查看购物车、新书入库、封面上传和库存管理。所对应的页面如表 7-6 所示。

表 7-6　页面表

功　能	页　面	功　能	页　面
首页	Default. aspx	放入购买车	Purchase. aspx
用户登录	Login. aspx	查看购物车	ShoppingCart. aspx
用户注册	Register. aspx	新书入库	load_new. aspx
按名搜索	Search. aspx	封面上传	loadCover. aspx
查看图书介绍	ShowDetail. aspx	库存管理	Inventory. aspx

7.2.3　首页

1. 概述

首页具有导航功能，通过超链接链接到其他页面，如登录、查询、购物车、注册、新书入库、封面上传、库存管理等，如图 7-1 所示。同时具有图书推荐功能，单击推荐图书的图书封皮图片或图书名超链接，可进入该图书的介绍页面。



图 7-1　首页

2. 代码

```
<%@ Page Language = "C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat = "server"></script>
<html><head><title>网上书店首页</title>
<meta http-equiv = "Content-Type" content = "text/html; charset = utf-8" /></head>
<body alink = "#ffffff" link = "#ffff00" vlink = "yellow">
<table width = "80%" align = "center" cellspacing = "0"><tr>
<td colspan = "2" bgcolor = "#000000" style = "height: 60px">
<table width = "100%">
```

【小提示】

上述代码“<title>网上书店首页</title>”为设置网页标题；“colspan="2"”为合并两个单元格。

```
<tr><td colspan = "3" align = "center" style = "height: 45px; font-size:36px;
color: #0000ff; font-family:'幼圆'">
<blink><b>网上书店</b></blink></td></tr><tr>
<td width = "4%"> </td>
<td align = "center" style = "color:yellow; font-family: 幼圆; height: 21px;">★
<a href = " Login.aspx">登录</a>★
<a href = "Search.aspx">查询</a>★
<a href = "ShoppingCart.aspx">购物车</a>★
<a href = "orders.aspx">客户订单</a>★
<a href = "Register.aspx">注册</a>★</td>
<td width = "4%"> </td></tr></table></td></tr><tr>
<td colspan = "2" bgcolor = "#999999" style = "height: 21px"> </td></tr>
```

【小提示】

上述代码“登录”为设置超链接，超链接文字为“登录”，链接页面为登录页面 Login.aspx。

```
<tr><td width = "24%" align = "center" bgcolor = "#999999" style = "color: #0000ff;
font-family: 幼圆; height: 20px;">↓后台管理↓</td>
<td width = "76%" align = "center" bgcolor = "#999999" style = "color: #0000ff;
font-family: 幼圆; height: 20px;">↓推荐图书↓</td></tr><tr>
<td height = "15" colspan = "2" align = "center" bgcolor = "#999999"><hr/></td></tr><tr>
<td align = "center" valign = "top" bgcolor = "#999999">
<table width = "65%" align = "center" bgcolor = "#999999"><tr>
<td align = "center" style = "width: 115px"><a href = "load_new.aspx">
<span style = "color:yellow; font-family: 幼圆; height: 21px;">☆</span>新书入库</a>
</td></tr>
```


【小提示】

上述代码“<span style="color:yellow; font-family:

幼圆；height：21px;">☆新书入库"为设置超链接，超链接文字为"新书入库"，链接页面为新书入库页面 load_new.aspx；用<span>标记设置行内元素为黄色，字体为幼圆，高 21 像素。

```
<tr><td align = "center" style = "width: 115px">
<a href = "loadCover.aspx">
<span style = "color:yellow; font - family: 幼圆; height: 21px;">☆</span>封面上传</a>
</td></tr><tr>
<td align = "center" style = "width: 115px">
<a href = "inventory.aspx">
<span style = "color:yellow; font - family: 幼圆; height: 21px;">☆</span>库存管理</a>
</td></tr></table></td>
<td width = "76 % " valign = "top" bgcolor = " # cacaca">
<table width = "100 % " cellspacing = "0"><tr>
<td width = "33 % " bgcolor = " # cacaca" style = "height: 206px">
<table width = "28 % " border = "0" align = "center" cellspacing = "0">
<tr><td width = "33 % ">
<a href  = "ShowDetail.aspx book_id = 601" >
<img src = "BookImages/601.gif" width = "120" height = "155" /></a>
</td></tr><tr>
<td align = "center" style = "height: 21px">
<a href  = "ShowDetail.aspx book_id = 601" > ASP.NET 开发实战宝典</a>
</td></tr></table>
```

【小提示】

上述代码"ASP.NET 开发实战宝典"为设置超链接，超链接文字为"ASP.NET 开发实战宝典"，链接页面为查看图书介绍页面 ShowDetail.aspx，传递参数 book_id 值为《ASP.NET 开发实战宝典》这本书的图书编号 601。

```
</td><td width = "33 % " bgcolor = " # cacaca" style = "height: 206px">
<table width = "20 % " border = "0" align = "center"><tr>
<td width = "34 % ">
<a href  = "ShowDetail.aspx book_id = 602" >
<img src = "BookImages/602. gif" width = "120" height = "155 /></a>
</td></tr><tr>
<td align = "center"><a href  = "ShowDetail.aspx book_id = 602" > C# 开发实战宝典</a>
</td></tr></table>
```

【小提示】

上述代码""为在页面上显示图片，图片为 BookImages 文件夹中的 602.gif，宽和高分别为 120 像素和 155 像素。

```
</td><td width = "34 % " bgcolor = " # cacaca" style = "height: 206px">
<table width = "41 % " align = "center" cellspacing = "0"><tr>
<td width = "34 % "><a href  = "ShowDetail.aspx book_id = 603" >
<img src = "BookImages/603. gif" width = "120" height = "155" /></a>
</td></tr><tr>
<td align = "center"><a href  = "ShowDetail.aspx book_id = 603" > JAVA 开发实战宝典</a>
```

```
</td></tr></table></td></tr><tr><td bgcolor="#cacaca">
<table width="47%" border="0" align="center"><tr>
<td><a href ="ShowDetail.aspx book_id=604" >
<img src="BookImages/604.gif" width="120" height="155" /></a>
</td></tr><tr><td align="center">
<a href ="ShowDetail.aspx book_id=604" >PHP开发实战宝典</a>
</td></tr></table></td>
<td bgcolor="#cacaca">
<table width="37%" border="0" align="center"><tr>
<td width="57%">
<a href ="ShowDetail.aspx book_id=605" >
<img src="BookImages/605.gif" width="120" height="155" /></a>
</td></tr><td align="center">
<a href ="ShowDetail.aspx book_id=605" >Visual Basic开发实战宝典</a>
</td></tr></td><td bgcolor="#cacaca">
<table width="24%" border="0" align="center">
<tr><td><a href ="ShowDetail.aspx book_id=606" >
<img src="BookImages/606.gif" width="120" height="155" /></a>
</td></tr><tr><td align="center">
<a href ="ShowDetail.aspx book_id=606" >Visual C++开发实战宝典</a>
</td></tr></table></td></tr></table></td></tr><tr>
<td height="30" colspan="2" align="center" bgcolor="#999999"><hr/></td>
</tr></table></body></html>
```

7.2.4 用户管理

1. 用户登录

1）概述

在简单网上书店系统中，用户进行查看图书或购书，必须以会员身份登录。用户登录页面如图 7-2 所示。



图 7-2 用户登录页面

2）代码

```
<%@ Import Namespace ="System.Data " %>
<%@ Import Namespace="System.Data.SqlClient" %>    //导入命名空间
<%@ Page Language="C#" Debug ="true"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script Language="C#" runat="server" >
void submitbtn_click(object sender, EventArgs E)        //单击登录按钮代码
{
    SqlConnection conn = new
    SqlConnection((string)ConfigurationSettings.AppSettings["connString"]);
    //创建数据库连接对象 SqlConnection
    String queryStr = "select * from users where user_id='" + Request.Form["UserID"] + "'";
    //在用户表中按用户编号查询
    SqlDataAdapter userAdapter = new SqlDataAdapter(queryStr, conn);
                                                        //创建适配器对象 SqlDataAdapter
    DataSet userSet = new DataSet();                    //创建数据集对象 DataSet 对象
    userAdapter.Fill(userSet, "users");                 //对数据集对象 DataSet 进行填充
    if (userSet.Tables["users"].Rows.Count == 0)
    {
        Message.Text = "没有这个用户,请重新输入";
        return;
    }
    if (Userpassword.Value != userSet.Tables["usres"].Rows[0]["user_pwd"].ToString())
    {
        Message.Text = "密码输入错误,请重新输入";
        return;
    }
    FormsAuthenticationTicket ticket = new FormsAuthenticationTicket(UserID.Value,
    false, 1800);
    FormsAuthentication.RedirectFromLoginPage(UserID.Value, false);
    return;
}
</script>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>用户登录</title>
<style type="text/css">
.style1
{width: 45px;}
</style>
</head>
<body bgcolor =" #CACACA" text =" #FFFFFF" link =" #FFFF00" vlink =" #FFFF00"
alink =" #00FF00">
<div align =" Center"  style =" width:762;  height:250 ">
<h1><font face="幼圆 "  color="#0000FF">
          网上书店会员登录
```

```
            </font></h1>
<form id="Form2"  method = "post" runat = "server" align = "Center"><br>
<h3><font face="幼圆"  color="#0000FF">
            
     请输入你的会员账号和密码:</font></h3>
<div align =" center"><center>
<table cellspacing = "0"  cellpadding = "2"  bgcolor=" #FFFFFF"
style="border-collapse :collapse " bordercolor =" #FFFFFF"><tr>
<td class="style1" ><font color=" #0000FF" face="幼圆">账号:</font></td>
<td><input type="text" id="UserID" runat="server" size="20"/></td>
</tr><tr><td class="style1" ><font color="#0000FF" face="幼圆">密码:</font></td>
<td><input type="password" id="Userpassword" runat="server" size="20"/></td></tr>
```

【小提示】

上述代码“<input type="password" id="Userpassword" runat="server" size="20"/>”为定义密码框。

```
<tr>
<td class="style1" ></td>
<td>      
<asp:Button id = "SubmitBtn" OnClick = "SubmitBtn_Click" runat = "server"
Text = "登录" Height="26px" Width="74px"/>
           
<input type = "reset" value = "清除" style="width: 65px">
         
<asp:Label id = "Message" ForeColor = " red " runat="server"/>
</td>
</tr>
<tr><td colspan = "2" ><br></td></tr>
<tr><td colspan = "2" ><div><font face=" 幼圆" color = " #0000FF">没有会员账号 请
</font><a href=" Register.aspx" ><font color="Red">注册</font></a>
<font face=" 幼圆" color=" #0000FF">一个账号吧! </font></div></td></tr>
</input></table></center></form></div></body></html>
```

代码首先根据用户输入的用户名进行搜索,将用户信息读入 DataSet。如果 DataSet 的 users 表中行数为 0,表示没有这个用户名,则提示用户重新输入。

```
if (userSet.Tables["users"].Rows.Count == 0)
{
    Message.Text = "没有这个用户,请重新输入";
    return;
}
```

如果找到这个用户,就将用户输入的密码和 users 表的 user_pwd 字段中的密码做比较。如果不一致,就提示用户重新输入密码。

```
if (Userpassword.Value != userSet.Tables["usres"].Rows[0]["user_pwd"].ToString())
{
    Message.Text = "密码输入错误,请重新输入";
```

```
    return;
}
```

在通过用户名和密码的检查后，创建一个 FormsAuthenticationTicket 对象，并将用户重定向回首页。

```
FormsAuthenticationTicket ticket = new FormsAuthenticationTicket(UserID.Value, false,
1800);
FormsAuthentication.RedirectFromLoginPage(UserID.Value, false);
```

2. 用户注册

1）概述

用户注册后才可以通过登录进入网上书店进行操作。用户注册页面如图 7-3 所示。



图 7-3 用户注册页面

2）代码

```
<%@ Import Namespace = "System.Data" %>
<%@ Import Namespace = "System.Data.SqlClient" %>
<%@ Import Namespace = "System.Web.Mail" %>
<%@ Page Language = "C#" Debug = "true" %>
<script language = "c#" runat = "server">
void Button1_Click(Object sender, EventArgs E)
{
    SqlConnection conn = new SqlConnection
    ((String)ConfigurationSettings.AppSettings ["connString"]);
    String insertCmd = "insert into users(user_id,user_name,user_pwd,user_email)
    values(@user_id,@user_name,@user_password,@email)";
    SqlCommand istCmd = new SqlCommand(insertCmd,conn);
    istCmd.Parameters .Add("@user_id",SqlDbType.VarChar,50).Value = user_id.Text;
    istCmd.Parameters .Add("@user_name",SqlDbType.VarChar,20).Value = user_name.Text;
    istCmd.Parameters .Add("@user_password",SqlDbType.VarChar,20).Value = user_
    password.Text;
    istCmd.Parameters .Add("@email",SqlDbType.VarChar,50).Value = email.Text;
    try
```

```
        {
            conn.Open();                        //打开连接
            istCmd.ExecuteNonQuery();        //以上 11 行代码将用户注册信息插入用户表 users
            conn.Close();                       //关闭连接
            MailMessage myMail = new MailMessage();//创建邮件对象 MailMessage
            myMail.BodyFormat =  MailFormat.Html;  //邮件格式
            myMail.Subject = "网上书店会员注册";  //邮件主题
            myMail.From = "yx123456789@126.com"; //邮件发送地址
            myMail.To = email.Text;             //邮件目的地址
            myMail.Body = "您已经成功注册成为网上书店的会员!<br>您的账号是:" +
            user_id.Text + "<br>您的密码是:" + user_password.Text + "<p/>";//邮件内容
            SmtpMail.SmtpServer  =  "127.0.0.1";                    //邮件服务器地址
            SmtpMail.Send(myMail);                                  //发送邮件
            Response.Redirect("Default.aspx");              //重定向到主页 Default.aspx
        }
        catch(System.Data .SqlClient.SqlException e)
        {
            if(e.Number == 2627)
            {
                RequiredFieldValidator2.ErrorMessage = "用户编号已存在请重新输入!";
                RequiredFieldValidator2.IsValid = false;
            }
            Response.Write(e.ToString());
        }
    } </script>
    <!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - transitional.dtd">
    <html xmlns = "http://www.w3.org/1999/xhtml" >
    <head runat = "server">
    <head><title>会员注册</title>
    <meta http - equiv = "Content - Type"content = "text/html;charset = big5"/></head>
    <body bgcolor = " # CACACA" text = " # FFFFFF" link = " # FFFF00" vlink = " # FFFF00" alink =
    " # 00FF00">
    <center><h1 ><font color = " # 000000">网上书店会员注册</font ></h1 ></center >
    <form method = "post"  id = "form"  runat = "server">
    <table width = "400"border = "0"cellspacing = "2"align = "Center">
    <tr><td colspan = "2">
    <div align = "center"><font size = "4"color = " # 0000FF"><
    <font face = "宋体">您的个人资料</font >></font >
    <font size = "4">       <font color = "red"> *
    </font ></font ><font class = "font9ptblack"face = "宋体"color = " # 0000FF">
    <font size = "4">表示必填</font ></font ></div >
    </td ></tr ><tr ><td valign = "top"align = "right">
    <font color = " # 0000FF"><font face = "宋体">姓名</font ></td >
```

【小提示】

上述代码“<您的个人资料>”中设置“您的个人资料”的字体为宋体，字号为 4，颜色为蓝色。

```
    <td><asp:TextBox id = "user_name"  columns = "20"  maxlength = "50"  runat = "server"/>
    <font color = "red"> * </font ><br/>
    <asp:RequiredFieldValidator id  = "RequiredFieldValidator1"  ControlToValidate = "user_name"
```

```
Display = "Dynamic" ErrorMessage = "请输入姓名!" runat = "server"/></td></tr><tr>
<td valign = "top"align = "right">
<font color = "#0000FF">用户编号</font></td><td>
<asp:TextBox id = "user_id" columns = "20" runat = "server"/><font color = "red"> * </font>
<br/>
<asp:RequiredFieldValidator id = "RequiredFieldValidator2" ControlToValidate = "user_id"
Display = "Dynamic" ErrorMessage = "请输入用户编号!" runat = "server"/></td></tr><tr>
<td valign = "top"align = "right">
<font color = "#0000FF"><font face = "宋体">密码</font></font></td><td>
<asp:TextBox id = "user_password" TextMode = "Password" columns = "20" maxlength = "20"
runat = "server"/>
<font color = "red"> * </font><br/>
< asp: RequiredFieldValidatorDisplay = " Dynamic " id = " RequiredFieldValidator3 "
ControlToValidate = "user_password" ErrorMessage = "请输入密码!" runat = "server"/></td>
</tr>
```

【小提示】

上述代码“< asp: RequiredFieldValidatorDisplay ="Dynamic" id ="RequiredFieldValidator3" ErrorMessage ="请输入密码!" runat ="server"/>”为必填验证，如果验证未通过，则显示“请输入密码!”的错误提示信息。

```
<tr><td valign = "top"align = "right">
<font color = "#0000FF"><font face = "宋体">确认密码</font></font></td><td>
<asp:TextBox id = "user_password1" TextMode = "Password" columns = "20" maxlength = "20"
runat = "server"/>
<font color = "red"> * </font><br/>
<asp:RequiredFieldValidator id = "RequiredFieldValidator4" ControlToValidate =
"user_password1" Display = "Dynamic" ErrorMessage = "请确认密码!id!" runat = "server"/>
<asp:CompareValidator id = "CompareValidator1" ControlToValidate = "user_password"
controltocompare = "user_password1" Type = "String" Operator = "Equal" Display = "Dynamic"
ErrorMessage = "两次输入密码不相同,请重新输入!" runat = "server"/></td></tr>
```

【小提示】

上述代码“< asp: CompareValidator id ="CompareValidator1" ControlToValidate ="user_password" controltocompare ="user_password1" Type ="String" Operator ="Equal" Display="Dynamic" ErrorMessage ="两次输入密码不相同，请重新输入!" runat="server"/>”为比较验证，如果验证未通过，则显示“两次输入密码不相同，请重新输入!”的错误提示信息。

```
<tr><td valign = "top"align = "right">
<font color = "#0000FF"> E - mail </font></td><td>
<asp:TextBox id = "email"columns = "30" maxlength = "50" runat = "server"/>
<font color = "red"> * </font><br/>
<asp:RequiredFieldValidator id = "RequiredFieldValidator6" ControlToValidate = "email"
Display = "Dynamic" ErrorMessage = "请输入 E - mail! id!" runat = "server"/>
<asp:RegularExpressionValidatorDisplay = "Dynamic"id = "RegularExpressionValidator1"
runat = "server"ControlToValidate = "email"Display = "Dynamic"
ValidationExpression = "\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*"
```

```
ErrorMessage = "E - mail 格式输入错误!"></asp:RegularExpressionValidator ></td ></tr >
```



【小提示】

上述代码“< asp: RegularExpressionValidator Display = " Dynamic " id = "RegularExpressionValidator1" runat = " server " ControlToValidate = " email " ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*" ErrorMessage="E-mail 格式输入错误!"></asp:RegularExpressionValidator>”为格式验证，如果验证未通过，则显示“E-mail 格式输入错误!”的错误提示信息。

```
<tr align = "center">< td colspan = "2">
<asp:Button id = "Buttion1" Text = "注册" OnClick = "Button1_Click"  runat = "server"/>
</td ></tr ></table ></form ></body ></html >
```

这段代码中，接收用户输入的信息，使用 ExecuteNonQuery()方法将其插入 users 表中。然后使用 MailMessage 对象给用户发送注册成功的信息。

在验证用户的输入时，使用了 ASP.NET 的验证控件。

RequiredFieldValidator：控件验证必填项是否有输入。

CompareValidator：控件验证两次输入的密码是否一致。

RegularExpressionValidator：控件验证 E-mail 格式是否输入正确。

7.2.5 图书管理

1. 新书入库

1）概述

新书入库页面在录入新书时，图书编号不允许重复，类型可以从下拉列表框中选择。“新书入库”页面如图 7-4 所示。

图 7-4 “新书入库”页面

2）代码

```
<%@ Page Language="C#" %>
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.SqlClient" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script language="C#" runat="server">
//以下为添加按钮的单击事件代码
private void Add_Click(Object sender, EventArgs E)
{
    SqlConnection conn = new SqlConnection((string)
    ConfigurationSettings.AppSettings["connString"]);
    string queryStr = "insert into books (book_id,title,type,price,notes,inventory) values
    (@book_id,@title,@type,@price,@notes,@inventory)";
    SqlCommand addCmd = new SqlCommand(queryStr, conn);
    addCmd.Parameters.Add("@book_id", SqlDbType.VarChar, 6).Value = book_id.Text;
    addCmd.Parameters.Add("@title", SqlDbType.VarChar, 80).Value = title.Text;
    addCmd.Parameters.Add("@type", SqlDbType.VarChar, 20).Value = type.SelectedItem.Text;
    addCmd.Parameters.Add("@price", SqlDbType.Money, 8).Value = price.Text;
    addCmd.Parameters.Add("@inventory", SqlDbType.Int, 4).Value = inventory.Text;
    addCmd.Parameters.Add("@notes", SqlDbType.VarChar, 250).Value = notes.Text;
    try
    {
        conn.Open();
        addCmd.ExecuteNonQuery();        //以上 13 行代码将新书信息插入图书信息表 books
        conn.Close();
        Msg.Text = "新书已入库!";
        ClearText();
    }
    catch (System.Data.SqlClient.SqlException e)
    {
        if (e.Number != 0)
        {
            Msg.Style["color"] = "red";
            if (e.Number == 2627)
            {
                Msg.Text = "图书编号重复,请重新输入!";
            }
            else
            Msg.Text = "ERROR:" + "[" + e.Number.ToString()+"]" + e.ToString();
        }
    }
}
//以下为 ClearText 函数代码,用于清空文本框
private void ClearText()
{
    book_id.Text = "";
    title.Text = "";
    price.Text = "";
```

```
        inventory.Text = "";
        notes.Text = "";
    }
    //以下为退出添加按钮的单击事件代码
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Redirect("default.aspx");    //重新定位到主页 default.aspx
    }
    </script>
    <html xmlns = "http://www.w3.org/1999/xhtml" >
    <head runat = "server">
    <title>新书入库</title>
    </head>
    <body bgcolor = "#CACACA" text = "#023456" vlink = "#FFFF00" alink = "#00FF00">
    <h1 ><font face = "幼圆" size = "5" color = "#000080">新书入库</font></h1 >
    <form id = "form1" runat = "server">
    <font color = "#000080"><font face = "幼圆">图书编号:
    <asp:TextBox ID = "book_id" text = "" MaxLength  = "6" runat = "server" /></font>
    <br />
    <font face = "幼圆"   color = "#000080">书名:
    <asp:TextBox ID = "title" text = "" MaxLength  = "50" runat = "server" /></font>
    <br />
    <font face = "幼圆"    color = "#000080">类型:
    <asp:DropDownList ID = "type" runat = "server">
    <asp:ListItem > NET </asp:ListItem >
    <asp:ListItem > CPP </asp:ListItem >
    <asp:ListItem > LINUX </asp:ListItem >
    <asp:ListItem > JAVA </asp:ListItem >
    <asp:ListItem > SE </asp:ListItem >
    </asp:DropDownList ></font ></font ><p />
```

【小提示】

上述代码“<asp:DropDownList ID="type" runat="server"><asp:ListItem>NET</asp:ListItem><asp:ListItem>CPP</asp:ListItem><asp:ListItem>LINUX</asp:ListItem><asp:ListItem>JAVA</asp:ListItem><asp:ListItem>SE</asp:ListItem></asp:DropDownList>"为利用下拉列表框控件进行图书类型选择。

```
    <font face = "幼圆"    color = "#000080">价格:
    <asp:TextBox ID = "price" Text  = "" MaxLength  = "8" runat = "server"/></font ><p />
    <font face = "幼圆"    color = "#000080">存库量:
    <asp:TextBox ID = "inventory" Text = "" MaxLength  = "4" runat = "server"/></font ><p />
    <font face = "幼圆"    color = "#000080">说明:</font ><br/>
    <asp:TextBox ID = "notes" Text  = "" MaxLength  = "200" TextMode = "MultiLine" Columns = "40"
    Rows  = "4" runat = "server" /><p />
    <asp:Button ID  = "Add" Text  = "添加" OnClick  = "Add_Click" runat = "server" />
    <asp:Button ID = "Button1" runat = "server" OnClick = "Button1_Click" Text = "退出添加" />
    <br />
```

```
<asp:Label ID = "Msg" Text = "" runat = "server" />
</form></body></html>
```

2. 封面上传

1）概述

因封面图片占的空间较大，图书封面存入数据库会影响数据库读取速度，因此利用封面上传页面将图书封面上传到网站的图片文件夹下。

为了根据图书编号选择一致的图书封面图片上传，本网上书店系统把封面图片文件名的主名定为图书的编号，扩展名为 gif。例如，《C# 开发实战宝典》的图书编号为 102，它的封面图片文件名就为 102.gif。封面上传页面如图 7-5 所示。

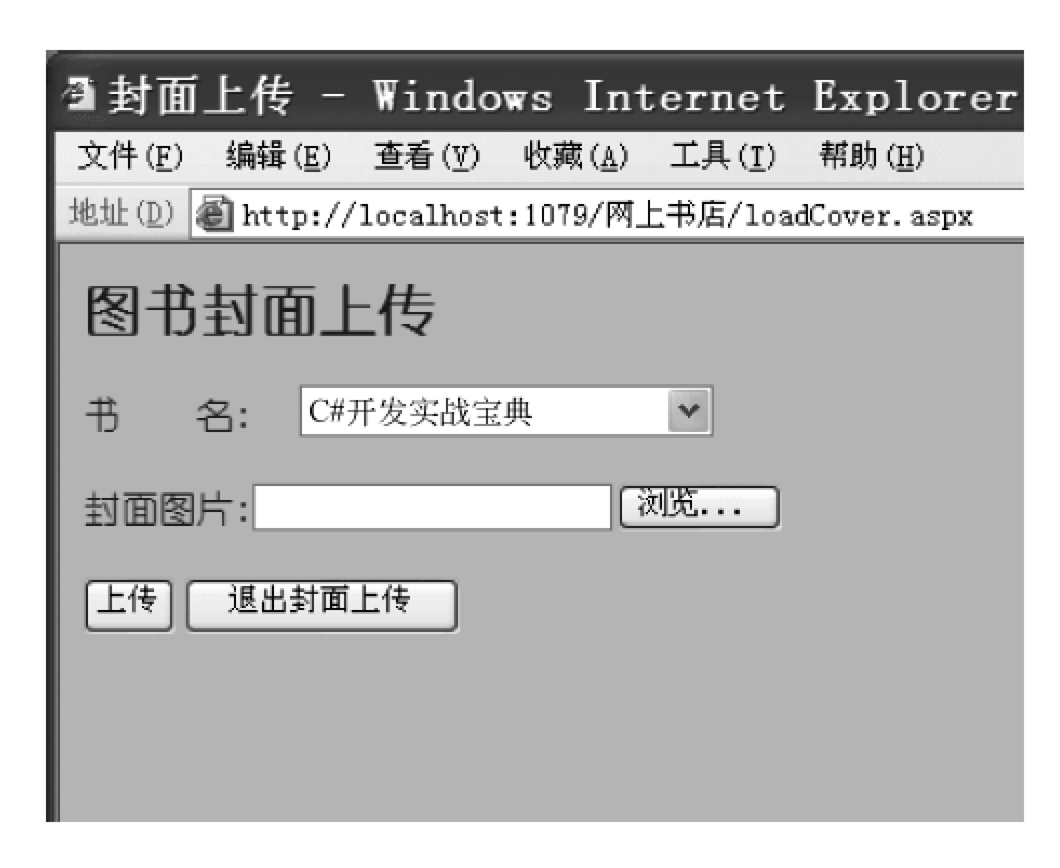


图 7-5 封面上传页面

2）代码

```
<%@ Import Namespace = "System.Data" %>
<%@ Import Namespace = "System.Data.SqlClient" %>
<%@ Page Language = "C#" Debug = "true" %>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1 - transitional.dtd">
<script runat = "server">
void Page_load(Object Source, EventArgs E)
{
    if (!Page.IsPostBack)
    {
        SqlConnection conn = new SqlConnection
        ((String)ConfigurationSettings.AppSettings["connString"]);
        SqlDataAdapter bookAdapter = new SqlDataAdapter ( " select book _ id, title from
        books", conn);
        //查询图书编号和书名
        DataSet bookSet = new DataSet();
        bookAdapter.Fill(bookSet, "books");
        bookList.DataSource = bookSet.Tables["books"].DefaultView;
        bookList.DataTextField = "title";
        bookList.DataValueField = "book_id";
```

```
            bookList.DataBind();
        }
    }
    //以下为上传按钮单击事件代码
    void UpLoad_Click(Object sender,EventArgs E)
    {
        try
        {
            Request.Files["UpLoad"].SaveAs("BookImages/" + bookList.SelectedItem.Value +
            ".gif");
            //将封面图片存到 BookImages 文件夹中
            catch (Exception e)
            {
                Response .Write (e.ToString ());
            }
        }
        //以下为退出上传按钮单击事件代码
        protected void Button1_Click(object sender, EventArgs e)
        {
            Response.Redirect("default.aspx");
        }
    }
    </script>
    <html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server"><title>封面上传</title></head>
    <body bgcolor="#CACACA" text="#FFFFFF" vlink="#FFFF00" alink="#00FF00">
    <form id="form1" enctype="multipart/form-data" runat="server">
    <h1><font face="幼圆" size="5" color="#000080">图书封面上传</font></h1>
    <asp:Label ID ="Msg" runat="server" /><p />
    <font face="幼圆" color="#000080">书      名:  </font>
    <asp:DropDownList ID="bookList" runat="server" Width="176px" >
    </asp:DropDownList><p/>
    <font face="幼圆" color="#000080">封面图片:</font><input type ="file" id="UpLoad"
    runat="server" size="20" /><p />
    <input type ="submit" onserverclick ="UpLoad_Click" runat ="server" value ="上传"
    id="Submit1" />
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="退出封面上传"
    />
    </form></body></html>
```

3. 库存管理

1）概述

“库存管理”页面主要实现库存查看以及图书信息的修改。库存管理页面使用DataList服务器控件显示books表中的数据，如图7-6所示。单击“修改”超链接，可编辑该图书信息，如图7-7所示。

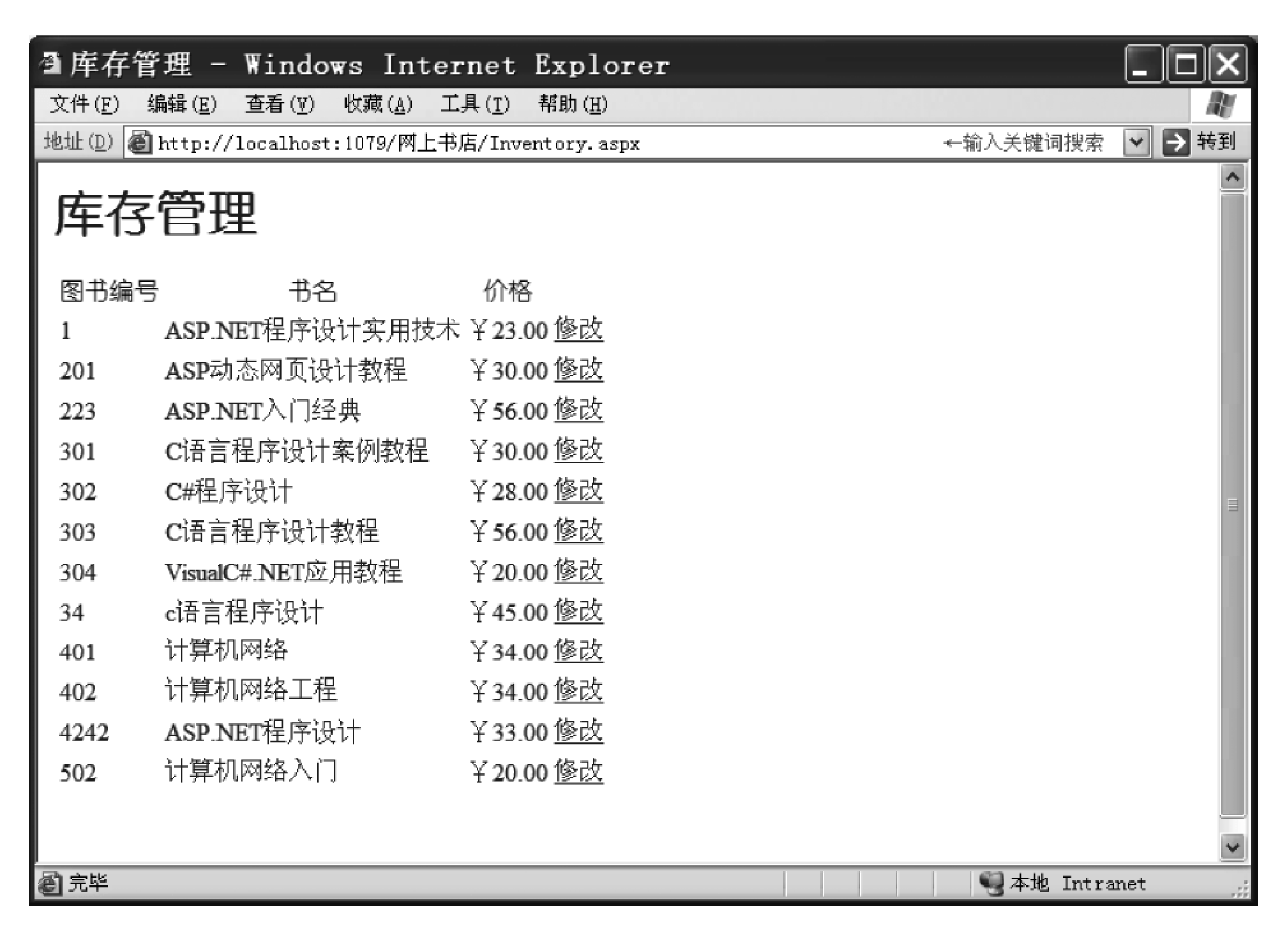


图 7-6 “库存管理”页面

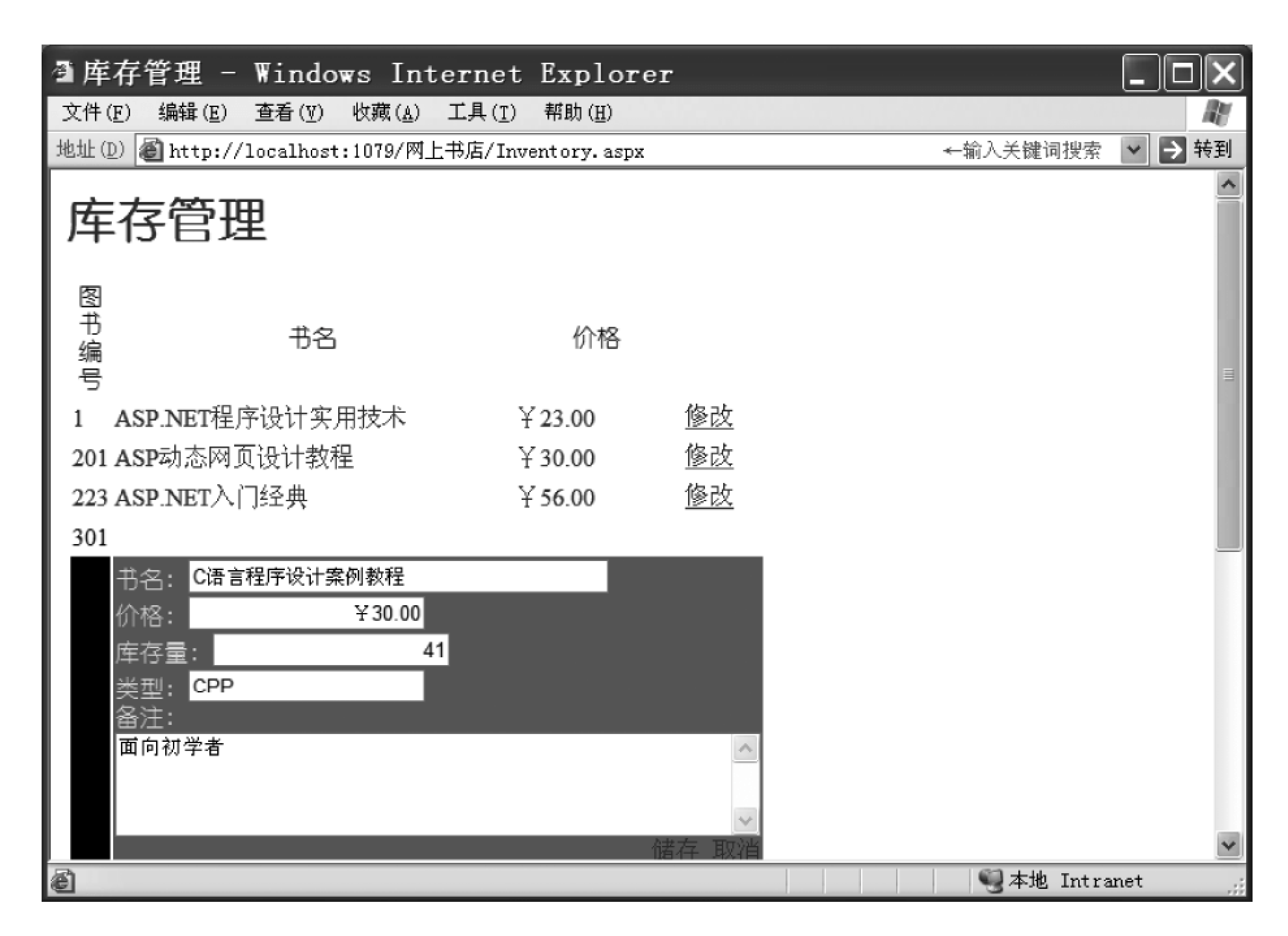


图 7-7 图书信息修改

2）代码

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
void Page_Load(Object semder, EventArgs e)
{
    if (!IsPostBack)
    BindList();
}
//以下为 BindList 函数代码,实现在页面上显示图书信息
void BindList()
{
    SqlConnection conn = new SqlConnection
    ((String)ConfigurationSettings.AppSettings["connString"]);
    SqlDataAdapter adapter = new SqlDataAdapter("select * from books", conn);
    DataSet ds = new DataSet();
    adapter.Fill(ds, "books");
    bookList.DataSource = ds.Tables["books"].DefaultView;
    bookList.DataBind();                    //绑定
}
//以下为处理"修改"事件的 bookList_EditCommand 函数代码
void bookList_EditCommand(Object Sender, DataListCommandEventArgs e)
{
    bookList.EditItemIndex = (int)e.Item.ItemIndex;
    BindList();
}
//以下为处理"取消"事件的 bookList_CancelCommand 函数代码
void bookList_CancelCommand(Object Sender, DataListCommandEventArgs e)
{
    bookList.EditItemIndex = -1;
    BindList();
}
//以下为处理"储存"事件的 bookList_UpdateCommand 函数代码
void bookList_UpdateCommand(Object Sender, DataListCommandEventArgs e)
{
    SqlConnection conn = new SqlConnection
    ((String)ConfigurationSettings.AppSettings["connString"]);
    String updateStr = "update books set
    title=@title,price=@price,inventory=@inventory,notes=@notes where book_id=
    @book_id";
    SqlCommand updateCmd = new SqlCommand(updateStr, conn);
    updateCmd.Parameters.Add("@book_id", SqlDbType.VarChar, 6).Value =
    ((Label)e.Item.FindControl("bookIdLabel")).Text;
    updateCmd.Parameters.Add("@title", SqlDbType.VarChar, 80).Value =
    ((TextBox)e.Item.FindControl("txtTitle")).Text;
    updateCmd.Parameters.Add("@price", SqlDbType.Money, 8).Value =
```

```
    ((TextBox)e.Item.FindControl("txtPrice")).Text;
    updateCmd.Parameters.Add("@inventory", SqlDbType.Int, 4).Value =
    ((TextBox)e.Item.FindControl("txtInventory")).Text;
    updateCmd.Parameters.Add("@notes", SqlDbType.VarChar, 250).Value =
    ((TextBox)e.Item.FindControl("txtNotes")).Text;
    try
    {
        conn.Open();
        updateCmd.ExecuteNonQuery();
        conn.Close();
        bookList.EditItemIndex = -1;
        BindList();
    }
    catch (System.Data.SqlClient.SqlException E)
    {
        if (E.Number != 0)
        {
            Response.Write(E.ToString());
        }
    }
}
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"><title>库存管理</title></head>
<body><h1 style="color: #0000ff; font-family: 幼圆">库存管理</h1>
<form id="form1" runat="server">
<asp:DataList ID="bookList" runat="server"
OnEditCommand ="bookList_EditCommand" OnUpdateCommand ="bookList_UpdateCommand"
OnCancelCommand ="bookList_CancelCommand"  Width="408px" Height="64px">
<HeaderTemplate>
<table><tr><td align ="center"><font face="幼圆">图书编号</font></td>
<td align ="center"><font face="幼圆">书名</font></td>
<td align ="center"><font face="幼圆">价格</font></td>
<td></td></tr></HeaderTemplate>
```

【小提示】

上述代码设置 DataList 编辑记录时执行的函数为 bookList_EditCommand，更新记录时执行的函数为 bookList_UpdateCommand，取消编辑记录时执行的函数为 bookList_CancelCommand。

```
<ItemTemplate><tr>
<td><font color="blue"><%#DataBinder.Eval(Container.DataItem,"book_id")%></font>
</td>
<td><font color="blue"><%#DataBinder.Eval(Container.DataItem,"title")%></font>
</td>
<td><font
color="blue"><%#DataBinder.Eval(Container.DataItem,"price","{0:c}")%></font></td>
<td><asp:LinkButton ID="button1" runat="server" Text ="修改" CommandName ="edit" />
```

```
</td>
</tr></ItemTemplate>
<EditItemTemplate><tr>
<td><asp:Label ID="bookIdLabel" runat="server" Text='<%#DataBinder.Eval
(Container.DataItem,"book_id")%>'/></td>
<td colspan ="2" align ="right" ></td></tr>
<tr style ="background-color:Gray">
<td style ="background-color :Black " valign ="top"></td>
<td colspan ="3"><font face="幼圆" color="#FFFF00">书名:
<asp:TextBox ID ="txtTitle" MaxLength ="20" Columns ="40" runat="server"
Text ='<%#DataBinder.Eval(Container.DataItem,"title")%>'/><br>价格:
<asp:TextBox ID ="txtPrice" Style="Text-Align:Right" MaxLength ="6" runat="server"
Text ='<%#DataBinder.Eval(Container.DataItem,"price","{0:c}")%>'/><br>库存量:
<asp:TextBox ID ="txtInventory" Style="Text-Align:Right" MaxLength ="4" runat="server"
Text ='<%#DataBinder.Eval(Container.DataItem,"inventory")%>'/><br>类型:
<asp:TextBox ID ="txtType" MaxLength ="20" runat ="server" Text ='<%#DataBinder.Eval
(Container.DataItem,"type")%>'/><br>备注:</font><br/>
<asp:TextBox ID ="txtNotes" MaxLength ="20" TextMode ="MultiLine" Rows ="4" Columns="50"
runat ="server" Text ='<%#DataBinder.Eval(Container.DataItem,"notes")%>'/>
<div align="right" >
<asp:LinkButton ID="button2" runat ="server" Text="储存" CommandName ="update" />
<asp:LinkButton ID="button3" runat ="server" Text="取消" CommandName ="cancel" />
</div></td></tr></table>
</EditItemTemplate>
</asp:DataList> 
</form></body></html>
```

4. 按名查询

1) 概述

按名查询页面可通过输入全部或部分书名进行图书查询,并且不区分大小写。按名查询页面如图 7-8 所示。输入书名后,单击"查询"按钮,可显示书名、类型、价格。查询结果页面如图 7-9 所示。单击查询结果页面要查看详细信息的图书书名,可进入查看图书介绍页面,查看该书的详细信息。

2) 代码

```
<%@ Import Namespace ="System.Data" %>
<%@ Import Namespace ="System.Data.SqlClient" %>
<script language ="C#" runat ="server">
void btnSearch_Click(Object sender,EventArgs E)
{
    SqlConnection sConn = new SqlConnection
    ((String) ConfigurationSettings. AppSettings["connString"]);
    string queryStr = "select book_id,title,price,type from books where title
    like '%" + keyText.Text + "%'";                //按书名进行模糊查询
    SqlDataAdapter resAdapter = new SqlDataAdapter(queryStr,sConn);
    DataSet  resSet = new DataSet();
    resAdapter.Fill(resSet,"books");
    bookGrid.DataSource = resSet.Tables ["books"].DefaultView;    //显示查询结果
```



图 7-8　“按名查询”页面



图 7-9　查询结果页面

```
        bookGrid.DataBind();
    }
    </script>
    <html>
    <meta http-equiv = "Content-Type" content = "text/html" />
    <head runat = "server">
    <title>按名查询</title>
    </head>
    <body bgcolor = "#0000ff" link = "Black" vlink = "#ff0000" alink = "#00ff00" >
    <h1 >查询结果如下...</h1 >
    <form id = "Form1" runat = "server">
    书名:
    <asp:TextBox id = "keyText"  maxlength = "50" runat = "server"/>
    <asp:Button id = "brnSearch" Text = "查询" onclick = "btnSearch_Click" runat = "server"/>
    </form>
    <asp:DataGrid id = "bookGrid" runat = "server" BorderColor = "black" BorderWidth = "2px"
    Font-Names = "Tohoma" GridLines = "Both" HeaderStyle-BackColor = "Gray"
    AutoGenerateColumns = "False" >
    <Columns>
    <asp:HyperLinkColumn HeaderText = "书名" DataNavigateUrlField = "book_id"
    DataNavigateUrlFormatString = "ShowDetail.aspx book_id = {0}" DataTextField = "title" />
    <asp:BoundColumn HeaderText = "类别" DataField = "type" />
    <asp:BoundColumn HeaderText = "价格" DataField = "price" DataFormatString = "{0:c}" >
    <ItemStyle HorizontalAlign = "Right" />
    </asp:BoundColumn>
    </Columns>
    <HeaderStyle BackColor = "#4A3C8C" Font-Bold = "True" ForeColor = "Black" />
    <FooterStyle BackColor = "#C6C3C6" ForeColor = "Black" />
    <SelectedItemStyle BackColor = "#9471DE" Font-Bold = "True" ForeColor = "White" />
    <PagerStyle BackColor = "#C6C3C6" ForeColor = "Black" HorizontalAlign = "Right" />
    <ItemStyle BackColor = "#DEDFDE" ForeColor = "Black" />
    </asp:DataGrid>  </body></html>
```

在本段代码中，显示结果使用了DataGrid控件，并在其中加入了一个超链接列HyperLinkColumn。

```
<asp:HyperLinkColumn HeaderText = "书名"
DataNavigateUrlField = "book_id"
DataNavigateUrlFormatString = "ShowDetail.aspx book_id = {0}"
DataTextField = "title" />
```

HeaderText属性指定列名。

DataNavigateUrlField属性获取数据源中要绑定到HyperLinkColumn中超链接URL的字段，这里绑定的字段是book_id，这样就可以根据图书编号来查询书籍的详细信息。

DataNavigateUrlFormatString属性指定HyperLinkColumn中超链接URL的显示格式，将其链接到图书介绍页面ShowDetail.aspx，并将参数book_id传给它。

DataTextField属性获取数据源中要绑定到HyperLinkColumn中超链接文本标题的字段。

另外，自定义了两列来显示图书类别和价格。

```
<asp:BoundColumn HeaderText = "类别" DataField = "type" />
<asp:BoundColumn HeaderText = "价格" DataField = "price" DataFormatString = "{0:c}" >
```

在定义价格列时，使用 DataFormatString 属性来设置指定列中各项字符串的显示格式，c 代表以货币格式显示数据。

可以在此基础上添加分页等新功能，以完善这个 DataGrid。

5. 查看图书介绍

1）概述

查看图书介绍页面，可以查看该书的编号、书名、类型、价格、封皮、说明等信息，如图 7-10 所示。单击页面右侧购物车图片，可以将该书放入购物车。



图 7-10　查看图书介绍页面

2）代码

```
<%@ Page Language = "C#" Debug = "true" %>
<%@ Import Namespace = "System.Data" %>
<%@ Import Namespace = "System.Data.SqlClient" %>
<script language = "c#" runat = "server">
//以下为页面载入事件代码
public void Page_Load(Object sender,EventArgs e)
{
    string bookID = (string)Request.QueryString["book_id"];
    SqlConnection conn = new SqlConnection
    ((String)ConfigurationSettings.AppSettings["connString"]);
    conn.Open();
    SqlDataReader dr = null;
    SqlCommand cmd = new SqlCommand("Select * From books where book_id = '" + bookID + "'",
    conn);
    dr = cmd.ExecuteReader();
    dr.Read();
    cover_img.Src = "imgs/" + dr.GetString(0) + ".jpg";
    book_id.Text = dr.GetString(0);
```

```
        title.Text = dr.GetString(1);
        type.Text = dr.GetString(2);
        price.Text = dr.GetSqlMoney(3).ToString();
        notes.Text = dr.GetString(4);
        purchaseBtn.NavigateUrl = "Purchase.aspx book_id = " + dr.GetString(0);
    }
    </script>
    <!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1 - transitional.dtd">
    <html xmlns = "http://www.w3.org/1999/xhtml" >
    <head runat = "server"><title>查看图书介绍</title></head>
    <body alink = "#00ff00" bgcolor = "#cacaca" link = "#ffff00" text = "#ffffff" vlink =
    "#ffff00">
    <table border = "0" width = "70%" align = "center">
    <tr><td rowspan = "4" width = "35%"><img  alt = "" id = "cover_img" runat = "server"
      src = "imgs/VisualC#.NET应用教程.jpg"/></td>
    <td width = "65%" style = "color: #0000ff; font - family: 幼圆">编号:
    <asp:Label ID = "book_id" runat = "server" /></td>
    </tr><tr><td width = "65%" style = "color: #0000ff; font - family: 幼圆">书名:
    <asp:Label ID = "title" runat = "server" /></td>
    </tr><tr><td width = "65%" style = "color: #0000ff; font - family: 幼圆">类型:
    <asp:Label ID = "type" runat = "server" /></td>
    </tr><tr><td width = "65%" style = "color: #0000ff; font - family: 幼圆">价格:
    <asp:Label ID = "price" runat = "server" /></td>
    </tr><tr><td colspan = "2" width = "65%" style = "color: #0000ff; font - family: 幼圆">
    说明:
    <blockquote><asp:Label ID = "notes" runat = "server" /></blockquote></td>
    </tr><tr><td colspan = "2" align = "right"><asp:HyperLink ID = "purchaseBtn"
    ImageUrl = "imgs/ shoppingcart.ico" runat = "server" /></td>
    </tr></table></body></html>
```

在这个页面中,使用 Request 对象的 QueryString()方法来获得传递过来的 book_id 参数,利用这个参数查询图书的详细信息。

```
string bookID = (string)Request.QueryString["book_id"];
```

因为查询结果只有一条记录,所以使用 DataReader 对象来接收查询结果。在页面上定义了 4 个 Label 服务器控件,将该记录的前 5 个字段的数据分别赋给它们的 Text 属性以在页面上显示出来。

```
book_id.Text = dr.GetString(0);
title.Text = dr.GetString(1);
type.Text = dr.GetString(2);
price.Text = dr.GetSqlMoney(3).ToString();
notes.Text = dr.GetString(4);
```

最后定义了一个 HyperLink 服务器控件,用来链接到放入购物车页面 Purchase.aspx,实现放入购物车功能。在它的 NavigateUrl 属性中,把当前图书的 book_id 作为参数传递给 Purchase.aspx 页面,以便在 Purchase.aspx 页面中使用。

```
purchaseBtn.NavigateUrl = "Purchase.aspx book_id = " + dr.GetString(0);
```

6．放入购物车

1）概述

实现将用户的购书信息添加到 ShoppingCart 表的功能。

2）代码

```
<%@ Import Namespace = "System.Data" %>
<%@ Import Namespace = "System.Data.SqlClient" %>
<%@ Page Language = "C#" %>
<!DOCTYPE html PUBLIC " - //W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1 - transitional.dtd">
<script runat = "server">
//以下为页面载入事件代码
protected void Page_Load(object sender, EventArgs e)
{
    SqlConnection conn = new
    SqlConnection((String)ConfigurationSettings.AppSettings["connString"]);
    SqlCommand cmd = new SqlCommand();          //创建 SqlCommand 命令对象 cmd
    cmd.Connection = conn;
    cmd.CommandType = CommandType.StoredProcedure;
    cmd.CommandText = "addToShoppingCart";      //将命令对象设置为执行存储过程
                                                //addToShoppingCart
    cmd.Parameters.Add("@user_id", SqlDbType.VarChar, 50).Value = Context.User.
    Identity.Name;
    string bookID = (string)Request.QueryString["book_id"];
    cmd.Parameters.Add("@book_id", SqlDbType.VarChar, 6).Value = bookID;
    try
    {
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
        Response.Redirect("ShoppingCart.aspx");
    }
    catch(System.Data.SqlClient.SqlException E)
    {
        Response.Write(e.ToString());
    }
}
</script><html xmlns = "http://www.w3.org/1999/xhtml" >
<head runat = "server"><title>放入购物车</title></head><body><form id = "form1" runat =
"server">
<div></div></form></body></html>
```

在这段代码中，使用了存储过程 addToShoppingCart 来实现将用户的购书信息添加到 ShoppingCart 表的功能。程序中给 Command 对象添加两个参数：@user_id 参数接收当前用户编号；@book_id 参数接收当前用户选中的图书编号。

```
cmd.Parameters.Add("@user_id", SqlDbType.VarChar, 50).Value = Context.User.Identity.Name;
string bookID = (string)Request.QueryString["book_id"];
cmd.Parameters.Add("@book_id", SqlDbType.VarChar, 6).Value = bookID;
```

Context.User.Identity.Name 用来获取当前用户编号，book_id 通过网页的链接语

句传过来。接收这两个参数后,使用 Command 对象的 ExecuteNonQuery()方法来执行存储过程。

存储过程 addToshoppingCart 的定义如下。

```
CREATE  PROCEDURE addToShoppingCart
@user_idVarChar(50), @book _idVarChar(6)
AS
Declare @Title VarChar(80),@Price money
if exists(select * from shoppingCart where book_id = @book_id)
begin
update shoppingCart set quantity = quantity + 1 where book_id = @book_id
end
else
begin
select @Title = title, @Price = price from books where book_id = @book_id
update shoppingCart set quantity = quantity + 1 where book_id = @book_id
insert into shoppingCart (user_id, book_id, title, quantity, price)
values(@user_id, @book_id, @Title,1, @Price)
end
```

该存储过程定义了两个参数@user_id 和@book_id,分别用来接收当前用户的编号和用户选中的图书编号;另外声明了两个变量@Title 和@Price,如果购物车表 ShoppingCart 中已经有所选图书,则将购物车表 ShoppingCart 中此书数量加 1。

```
if exists(select * from shoppingCart where book_id = @book_id)
begin
update shoppingCart set quantity = quantity + 1 where book_id = @book_id
end
```

否则从 books 表中找到用户所选图书,将此书的书名和价格分别赋给变量@Title 和@Price。

```
select @Title = title, @Price = price from books where book_id = @book_id
```

然后将用户编号(@user_id)、图书编号(@book_id)、书名(@Title)、数量(此时为 1)和价格(@Price)插入购物车表 ShoppingCart 中。

```
insert into shoppingCart (user_id, book_id, title, quantity, price)
values(@user_id, @book_id, @Title,1, @Price)
```

执行该存储过程后,将页面重定向到 ShoppingCart.aspx。

```
Response.Redirect("ShoppingCart.aspx");
```

7. 查看购物车

1) 概述

购物车页面可以查看所购图书、修改购书数量、结账或清空购物车。购物车页面如图 7-11 所示。页面中使用 DataList 控件来显示 shoppingCart 表中数据。为了便于修改,在 DataList 中定义了一个“修改数量”按钮列,用来修改购书数量。当单击每一行“修改数量”按钮时,该行的“数量”列将会出现一个编辑框,以便修改购书数量。同时“修改数

量”按钮变为“更新”和“取消”两个按钮。单击“更新”按钮，修改后的数量将被写入购物表 shoppingCart。单击“取消”按钮，则取消编辑，如图 7-12 所示。单击“清空购物车”按钮，从购物车表 shoppingCart 中删除该用户的信息，并显示“您的购物车是空的！”，如图 7-13 所示。



图 7-11 购物车页面



图 7-12 修改购书数量



图 7-13 清空购物车

2）代码

```
<%@ Import Namespace = "System.Data" %>
<%@ Import Namespace = "System.Data.SqlClient" %>
<%@ Page Language = "C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat = "server">
//以下为页面载入事件代码
protected void Page_Load(object sender, EventArgs e)
{
    if(!IsPostBack)
    BindList();
}
//以下为处理"修改数据"事件的 cartList_EditCommand 函数代码
void cartList_EditCommand(object sender, DataListCommandEventArgs e)
{
    cartList.EditItemIndex = (int)e.Item.ItemIndex;
    BindList();
}
//以下为 BindList 函数代码
void BindList()
{
    SqlConnection conn = new SqlConnection
    ((string)ConfigurationManager.AppSettings["connString"]);
    String userID = Context.User.Identity.Name;
    String queryStr = "select * from shoppingCart where user_id = '" + userID + "'";
    SqlDataAdapter cartAdapter = new SqlDataAdapter(queryStr, conn);
    DataSet cartSet = new DataSet();
```

```
    cartAdapter.Fill(cartSet,"shoppingCart");
    int count = cartSet.Tables["shoppingCart"].Rows.Count;
    if (count > 0)
    {
        cartList.DataSource = cartSet.Tables["shoppingCart"].DefaultView;
        cartList.Visible = true;
        divPanel.Visible = true;
        cartList.DataBind();
    }
    else{
        cartList.Visible = false;
        divPanel.Visible = false;
        cartMsg.InnerHtml = "您的购物车是空的!";
    }
}
//以下为"结账"按钮单击事件代码
private void purchase_Click(object sender, EventArgs E)
{
    SqlConnection conn = new SqlConnection
    ((string)ConfigurationManager.AppSettings["connString"]);
    SqlCommand purchaseCmd = new SqlCommand();
    purchaseCmd.Connection = conn;
    purchaseCmd.CommandType = CommandType.StoredProcedure;
    purchaseCmd.CommandText = "addNewOrder";
    purchaseCmd.Parameters.Add("@user_id", SqlDbType.VarChar, 50).Value =
    Context.User.Identity.Name;
    conn.Open();
    purchaseCmd.ExecuteNonQuery();
    conn.Close();
    Response.Redirect("default.aspx");
}
//以下为"清空购物车"按钮单击事件代码
private void clear_Click(object sender, EventArgs E)
{
    SqlConnection conn = new SqlConnection
    ((String)ConfigurationManager.AppSettings["connString"]);
    String deleteStr = "delete from shoppingCart where user_id = '" +
    Context.User.Identity.Name + "'";
    SqlCommand delCmd = new SqlCommand(deleteStr, conn);
    conn.Open();
    delCmd.ExecuteNonQuery();
    conn.Close();
    BindList();
}
//以下为处理"更新"事件的 cartList_UpdateCommand 函数代码
void cartList_UpdateCommand(object sender, DataListCommandEventArgs e)
{
    int quantity = Int16.Parse(((TextBox)e.Item.FindControl("quantityText")).Text);
    SqlConnection conn = new SqlConnection
```

```
        ((string)ConfigurationManager.AppSettings["connString"]);
        SqlCommand updateCmd = new SqlCommand();
        updateCmd.Connection = conn;
        if (quantity == 0)
        {
            string deleteStr = "delete shoppingCart where (user_id = @user_id and book_id =
            book_id_id)";
            updateCmd.CommandText = deleteStr;
        }
        else{
          string updateStr = "delete shoppingCart set quantity = @quantity where(user_id =
          @user_id and book_id = @book_id)";
          updateCmd.CommandText = updateStr;
          updateCmd.Parameters.Add("@quantity", SqlDbType.Int, 4).Value = quantity;
        }
        updateCmd.Parameters.Add("@user_id", SqlDbType.VarChar, 50).Value =
        Context.User.Identity.Name;
        updateCmd.Parameters.Add("@book_id", SqlDbType.VarChar, 6).Value =
        ((Label)e.Item.FindControl("bookIdLabel")).Text;
        conn.Open();
        pdateCmd.ExecuteNonQuery();
        conn.Close();
        cartList.EditItemIndex = -1;
        BindList();
    }
    //以下为处理"取消"事件的 cartList_CancelCommand 函数代码
    void cartList_CancelCommand(object sender, DataListCommandEventArgs e)
    {
        cartList.EditItemIndex = -1;
        BindList();
    }
    </script>
    <html xmlns = "http://www.w3.org/1999/xhtml" ><head runat = "server">
    <title>购物车</title></head><body><form id = "form1" runat = "server"><div>
    <asp:DataList ID = "cartList" runat = "server"  OnEditCommand = "cartList_EditCommand"
    OnUpdateCommand = "cartList_UpdateCommand"  OnCancelCommand = "cartList_CancelCommand">
    <HeaderTemplate><table><tr style = "background-color:Gray "><td>图书编号</td>
    <td>书名</td><td>价格</td><td>数量</td><td></td></tr></table>
    </HeaderTemplate>
    <ItemTemplate><tr><td><% # DataBinder.Eval(Container .DataItem ,"book_id") %></td>
    <td><% # DataBinder.Eval(Container .DataItem ,"title") %></td>
    <td><% # DataBinder .Eval(Container .DataItem ,"price","{0:c}") %></td>
    <td><% # DataBinder .Eval(Container .DataItem ,"quantity") %></td>
    <td><asp:LinkButton ID = "button1" runat = "server"  Text = "修改数量"
    CommandName = "edit" /></td></tr>
    </ItemTemplate>
    <EditItemTemplate>
    <tr style = "background-color:black">
```

```
<td><asp:Label ID = "bookIdLabel" runat = "server"
Text = '<% # DataBinder.Eval(Container.DataItem,"book_id") %>' /></td>
<td><asp:Label ID = "titleLabel" runat = "server"
Text = '<% # DataBinder .Eval (Container .DataItem ,"title") %>' /></td>
<td><asp:Label ID = "priceLabel" runat = "server"
Text = '<% # DataBinder .Eval (Container .DataItem ,"price","{0:c}") %>' /></td>
<td><asp:TextBox ID = "quantityText" Columns = "4" MaxLength = "4" runat = "server"
Text = '<% # DataBinder.Eval (Container .DataItem ,"quantity") %>' /></td>
<td><asp:LinkButton ID = "button2" runat = "server" Text = "更新" CommandName = "update" />
</td>
<asp:LinkButton ID = "button3" runat = "server" Text = "取消" CommandName = "cancel" />
</tr></EditItemTemplate></asp:DataList></div>
<h1 id = "cartMsg" runat = "server" ></h1>
<div id = "divPanel" align = "right" runat = "server" ></div>
<asp:Button ID = "purchase" Text = "结账" OnClick = "purchase_Click" runat = "server" />
<asp:Button ID = "clear" Text = "清空购物车" OnClick = "clear_Click" runat = "server" />
</form></body></html>
```

在“结账”按钮的单击事件代码中调用了一个存储过程 addNewOrder，将客户购书信息添加到客户订单状态表 orders 和客户订单详细信息表 orderDetail 表，并将用户编号 user_id 作为参数传递给该存储过程，该存储过程代码如下。

```
CREATE  PROCEDURE addNewOrder
@user_idVarChar(50)
AS
Declare @OrderId int;
set @OrderId = (select MAX(order_id) from orders where user_id = @user_id
insert into orders (order_id, user_id) values(@OrderId + 1, @user_id)
insert into orderDetails(order_id,book_id,title,quantity,price)
select @OrderId, book_id,title,quantity,price from shoppingCart where user_id = @user_id
delete shoppingCart where user_id = @user_id
```

该存储过程定义了一个参数@user_id 用来接收外部传来的用户编号，并声明了一个变量@OrderId 来获取该用户客户订单状态表 orders 的最大订单编号。

```
set @OrderId = (select MAX(order_id) from orders where user_id = @user_id
```

然后将@OrderId+1 和@user_id 插入客户订单状态表 orders 中。

```
insert into orders (order_id, user_id) values(@OrderId + 1, @user_id)
```

再将订单的详细信息加到订单详细信息表 orderDetail 中。

```
insert into orderDetail (order_id,book_id,title,quantity,price)
select @OrderId + 1, book_id,title,quantity,price from shoppingCart where user_id = @user_id
```

最后从购物车表 shoppingCart 中删除该用户的信息。

```
delete shoppingCart where user_id = @user_id
```

7.3 网站测试

网站测试指的是当一个网站制作完上传到服务器之后针对网站的各项性能情况的一项检测工作。它与软件测试有一定的区别，其除了要求外观的一致性以外，还要求其在各个浏览器下的兼容性以及在不同环境下的显示差异。主要内容如下。

1. 性能测试

（1）连接速度测试。用户连接到网站的速度与上网方式有关。

（2）负载测试。负载测试是在某一负载级别下，检测网站的实际性能，也就是能允许多少个用户同时在线。可以通过相应的软件在一台客户机上模拟多个用户来测试负载。

（3）压力测试。压力测试是测试网站的限制和故障恢复能力，也就是测试网站会不会崩溃。

2. 安全性测试

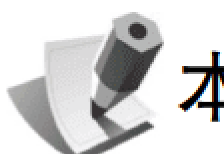
它需要对网站的安全性（服务器安全、脚本安全）、可能有的漏洞、攻击性及错误性进行测试；并对网站的服务器应用程序、数据、服务器、网络、防火墙等进行测试；用相对应的软件进行测试。

3. 基本测试

基本测试包括色彩的搭配，连接的正确性，导航的方便和正确，CSS应用的统一性。

4. 网站优化测试

好的网站是看它是否经过搜索引擎优化了，包括网站的架构、网页的栏目等。



本章小结

本章以网上书店为实例，从网站需求分析、网站总体设计、数据库设计、网站详细设计与开发进行了详细介绍，展示了网站开发的实现方法和基本流程。



思考与练习

自主创建一个网站。根据需求分析，进行数据库的设计和网站详细设计与开发。注意页面的规划和设计，尽量做到简洁大方、有吸引力。

附录 A　网站界面设计应遵循的几个原则

网站用户界面(Website User Interface)是指网站用于和用户交流的外观、部件和程序等。如果你经常上网,会看到很多网站设计很朴素,看起来给人一种很舒服的感觉;有些网站很有创意,能给人带来意外的惊喜和视觉的冲击;而相当多的网站页面上充斥着怪异的字体、花哨的色彩和图片,给人以网页制作粗劣的感觉。

网站界面的设计,既要从外观上进行创意以到达吸引眼球的目的,还要结合图形和版面设计的相关原理,从而使得网站设计变成了一门独特的艺术。通俗地讲,企业网站用户界面的设计应遵循以下几个基本原则。

1. 用户导向(User oriented)原则

设计网页首先要明确到底谁是使用者,要站在用户的观点和立场上来考虑设计网站。要做到这一点,必须要和用户沟通,了解他们的需求、目标、期望和偏好等。网页的设计者要清楚,用户之间差别很大,他们的能力各有不同。

比如有的用户可能会在视觉方面有欠缺(如色盲),对很多的颜色分辨不清;有的用户的听觉也会有障碍,对于网站的语音提示反应迟钝;而且相当一部分用户的计算机使用经验很初级,对于复杂一点的操作会感到很费力。

另外,用户使用的计算机配置也是千差万别,包括显卡、声卡、内存、网速、操作系统以及浏览器等都会有所不同。设计者如果忽视了这些差别,设计出的网页在不同的机器上显示就会造成混乱。

2. KISS(Keep It Simple and Stupid)原则

简洁和易于操作是网页设计的最重要的原则。毕竟,网站建设出来是用于普通网民来查阅信息和使用网络服务。没有必要在网页上设置过多的操作,堆集上很多复杂和花哨的图片。

该原则一般的要求:网页的下载不要超过10s(普通的拨号用户56Kbps网速);尽量使用文本链接,而减少大幅图片和动画的使用;操作设计尽量简单,并且有明确的操作提示;网站所有的内容和服务都在显眼处向用户予以说明等。

3. 布局控制

关于网页排版布局方面,很多网页设计者重视不够,网页排版设计的过于死板,甚至照抄他人。如果网页的布局凌乱,仅仅把大量的信息堆集在页面上,会干扰浏览者的阅读。一般在网页设计上所要遵循的原理有以下几个。

(1) Miller公式。根据心理学家George A. Miller的研究表明,人一次性接受的信息量在7b左右为宜。总结一个公式为:一个人一次所接受的信息量为(7±2)b。这一原理被广泛应用于网站建设中,一般网页上面的栏目选择最佳在5～9b之间,如果网站所提供给浏览者选择的内容链接超过这个区间,人在心理上就会烦躁、压抑,会让人感觉到信息

太密集，看不过来，很累。

例如，Aol. com 的栏目设置：Main、MyAol、Mail、People、Search、Shop、Channels 和 Devices 共 8 个分类。Msn. com 的栏目设置：MSN Home、My MSN、Hotmail、Search、Shopping、Money 和 People & Chat 共 7 项。然而很多国内的网站在栏目的设置远远超出这个区间。

(2) 分组处理。上面提到，对于信息的分类，不能超过 9 个栏目。

但如果内容实在是多，超出了 9 个，需要进行分组处理。如果，网页上提供几十篇文章的链接，需要每隔 7 篇加一个空行或平行线做以分组。如果网站内容栏目超出 9 个，如微软公司的网站，共有 11 个栏目，超过了 9 个，为了不破坏 Miller 公式，在设计时使用蓝、黑两种颜色分开，具体可以访问 www. microsoft. com。

4. 视觉平衡

网页设计时，各种元素(如图形、文字、空白)都会有视觉作用。根据视觉原理，图形与一块文字相比较，图形的视觉作用要大一些。所以，为了达到视觉平衡，在设计网页时需要以更多的文字来平衡一幅图片。另外，按照中国人的阅读习惯是从左到右、从上到下，因此视觉平衡也要遵循这个道理。

例如，很多的文字是采用左对齐〈Align=left〉，需要在网页的右面加一些图片或一些较明亮、较醒目的颜色。一般情况下，每张网页都会设置一个页眉部分和一个页脚部分，页眉部分常放置一些 Banner 广告或导航条，而页脚部分通常放置联系方式和版权信息等，页眉和页脚在设计上也要注重视觉平衡。

同时，也绝不能低估空白的价值。如果网页上所显示的信息非常密集，这样不但不利于读者阅读，甚至会引起读者反感，破坏该网站的形象。在网页设计上，适当增加一些空白，精炼网页，使得页面变得简洁。

5. 色彩的搭配和文字的可阅读性

颜色是影响网页的重要因素，不同的颜色对人的感觉有不同的影响。例如，红色和橙色使人兴奋并使得心跳加速；黄色使人联想到阳光，是一种快活的颜色；黑颜色显得比较庄重，考虑到希望对浏览者产生什么影响，为网页设计选择合适的颜色(包括背景色、元素颜色、文字颜色、链接颜色等)。

为方便阅读网站上的信息，可以参考报纸的编排方式将网页的内容分栏设计，甚至两栏也要比一满页的视觉效果要好。另一种能够提高文字可读性的因素是所选择的字体，通用的字体(Arial、Courier New、Garamond、Times New Roman、中文宋体)最易阅读，特殊字体用于标题效果较好，但是不适合正文。

如果在整个页面使用一些特殊字体(如 Cloister、Gothic、Script、Westminster、华文彩云、华文行楷)，这样读者阅读起来感觉一定很糟糕。该类特殊字体如果在页面上大量使用，会使得阅读颇为费力，浏览者的眼睛很快就会疲劳，不得不转移到其他页面。

6. 和谐与一致性

通过对网站的各种元素(颜色、字体、图形、空白等)使用一定的规格，使得设计良好的网页看起来应该是和谐的。或者说，网站的众多单独网页应该看起来像一个整体。网站

设计上要保持一致性,这又是很重要的一点。

一致的结构设计可以让浏览者对网站的形象有深刻的记忆;一致的导航设计可以让浏览者迅速而又有效地进入在网站中自己所需要的部分;一致的操作设计可以让浏览者快速学会在整个网站的各种功能操作。破坏这一原则,会误导浏览者,并且让整个网站显得杂乱无章,给人留下不良的印象。

当然,网站设计的一致性并不意味着刻板和一成不变,有的网站在不同栏目使用不同的风格,或者随着时间的推移不断改版网站,会给浏览者带来新鲜的感觉。

7. 个性化

1)符合网络文化

企业网站不同于传统的企业商务活动,要符合 Internet 网络文化的要求。首先,网络最早是非正式性、非商业化的,只是科研人员用来交流信息。其次,网络信息是只在计算机屏幕上显示而没有打印出来阅读,网络上的交流具有隐蔽性,谁也不知道对方的真实身份。另外,许多人在上网的时候是在家中或网吧等一些比较休闲、比较随意的环境下。

此时网络用户的使用环境所蕴含的思维模式与坐在办公室里西装革履的时候大相径庭。因此,整个互联网的文化是一种休闲的、非正式性的、轻松活泼的文化。在网站上使用幽默的网络语言,创造一种休闲的、轻松愉快、非正式的氛围会使网站的访问量大增。

2)塑造网站个性

另外,网站的整体风格和整体气氛表达要同企业形象相符合并应该很好地体现企业CI。在这方面比较经典的案例有:可口可乐个性鲜明的前卫网站 Life Tastes Good;工整、全面、细致的通用电气公司网站 We bring good things to life(GE 带来美好的生活);崇尚科技创新文化的 3M 公司网站 Creating solutions for business、industry and home;刻意扮演一个数字电子娱乐之集大成者的角色,要成为新时代梦想实现者的索尼网站;平易近人、亲情浓郁的通用汽车公司网站体现了“以人为本”的企业定位和营销策略;服务全面、细致、方便,处处体现“宾至如归”服务理念的希尔顿大酒店网站。

附录B　习 题 答 案

第 1 章习题答案

1. 选择题

(1) A　(2) D　(3) A　(4) B　(5) C

2. 问答题

(1) HTML 文档主要由 3 部分组成。HTML 部分以<HTML>标签开始，以</HTML>标签结束。头部以<HEAD>标签开始，以</HEAD>标签结束。主体部分包含在网页中显示的文本、图像和链接。主体部分以<BODY>标签开始，以</BODY>标签结束。

(2) 写出 URL 包含的 3 个部分内容的作用。

答：协议名称：是 WWW 服务器与客户之间遵循的通信协议。

服务器主机名称：用来标识该文件存储在哪个服务器上。

通信端口/文件目录/文件名称：是文件所在的目录路径和文件的名称。

(3) 简述动态网页的编程语言都有什么？它们的优缺点有哪些？

答：目前实现动态网页的技术主要有 ASP、ASP.NET、PHP 和 JSP。

ASP 语言的优点如下。

① 使用 VBScript、JScript 等简单易懂的脚本语言，结合 HTML 代码，即可快速地完成网站的应用程序。

② 无须 compile 编译，容易编写，可在服务器端直接执行。

③ 与浏览器无关(Browser Independence)，客户端只要使用可执行 HTML 码的浏览器，即可浏览 Active Server Pages 所设计的网页内容。

④ 能与任何 ActiveX Scripting 语言兼容。除了可使用 VBScript 或 JScript 语言来设计外，还通过 plug-in 的方式，使用由第三方所提供的其他脚本语言，如 REXX、Perl、Tcl 等。

⑤ 可使用服务器端的脚本来产生客户端的脚本。

⑥ ActiveX Server Components(ActiveX 服务器组件)具有无限可扩充性。

ASP 语言的缺点如下。

① Windows 本身的所有问题都会一成不变地也累加到了它的身上。安全性、稳定性、跨平台性都会因为与 Windows 的捆绑而显现出来。

② ASP 由于使用了 COM 组件，所以它会变得十分强大，但这样的组件或是操作中一不注意，那么外部攻击就可以取得相当高的权限而导致网站瘫痪或者数据丢失。

③ 由于 ASP 还是一种 Script 语言，所以除了大量使用组件外，没有办法提高其工作效率。

④ 无法实现跨操作系统的应用。

⑤ 无法完全实现一些企业级的功能：完全的集群、负载均衡。

JSP 语言的优点如下。

① 一次编写，随处运行。

② 系统的多平台支持。基本上可以在所有平台上的任意环境中开发，在任意环境中进行系统部署，在任意环境中扩展。

③ 强大的可伸缩性。从只有一个小的 Jar 文件就可以运行 Servlet/JSP，到由多台服务器进行集群和负载均衡，到多台 Application 进行事务处理、消息处理，一台服务器到无数台服务器，Java 显示了一个巨大的生命力。

④ 多样化和功能强大的开发工具支持。

⑤ 支持服务器端组件。Web 应用需要强大的服务器端组件来支持，开发人员需要利用其他工具设计实现复杂功能的组件供 Web 页面调用，以增强系统性能。JSP 可以使用成熟的 Java BEANS 组件来实现复杂商务功能。

JSP 语言的缺点如下。

① 与 ASP 也一样，Java 的一些优势正是它致命的问题所在。正是由于为了跨平台的功能，为了极度的伸缩能力，所以极大地增加了产品的复杂性。

② Java 的运行速度是用 class 常驻内存来完成的，所以它在一些情况下所使用的内存比较大。

PHP 语言的优点如下。

① 跨平台，性能优越，可以和很多免费的平台结合。

② 语法简单，如果有学习 C 和 Perl 的很容易上手，并且跟 ASP 有部分类似。

③ 有比较完整的支持，比如使用 ADODB 或者 PEAR::DB 做数据库抽象层，用 Smarty 或者 smart template 做模板层，如果是 PHP 5.1 的话，还能够使用 PDO(PHP Data Object)来访问数据库。

④ 有很多成熟的框架，比如支持 MVC 的框架 phpMVC，支持类似 ASP.NET 的事件驱动的框架 Prado，支持类似 Ruby On Rails 的快速开发的框架 Cake 等，足够满足你的应用需求。

⑤ 有很多开源的框架或开源的系统可以使用，比如比较知名的开源框架有 Zend Framework、CakePHP、CodeIgniter、symfony 等。

PHP 语言的缺点如下。

① 对多线程支持不太好，大多数时候我们只能简单的模拟去实现的。

② 语法不太严谨，比如变量不需要定义就可以使用，在 C，Java，C++ 中变量是必须先定义以后才可以使用的。

ASP.NET 语言的优点如下。

① 增强的性能。ASP.NET 是在服务器上运行的编译好的公共语言运行库代码。与被解释的前辈不同，ASP.NET 可利用早期绑定、实时编译、本机优化服务。这相当于在编写代码行之前便显著提高了性能。

② 世界级的工具支持。ASP.NET 框架补充了 Visual Studio 集成开发环境中的大

量工具箱和设计器。WYSIWYG 编辑、拖放服务器控件和自动部署只是这个强大的工具所提供功能中的少数几种。

③ 威力和灵活性。由于 ASP.NET 基于公共语言运行库,因此 Web 应用程序开发人员可以利用整个平台的威力和灵活性。.NET 框架类库、消息处理和数据访问解决方案都可从 Web 无缝访问。ASP.NET 也与语言无关,所以可以选择最适合应用程序的语言,或跨多种语言分割应用程序。另外,公共语言运行库的交互性保证在迁移到 ASP.NET 时保留基于 COM 的开发中的现有投资。

④ 简易性。ASP.NET 使执行常见任务变得容易,从简单的窗体提交和客户端身份验证到部署和站点配置。例如,ASP.NET 页框架使您可以生成将应用程序逻辑与表示代码清楚分开的用户界面,和在类似 Visual Basic 的简单窗体处理模型中处理事件。另外,公共语言运行库利用托管代码服务(如自动引用计数和垃圾回收)简化了开发。

⑤ 可管理性。ASP.NET 采用基于文本的分层配置系统,简化了将设置应用于服务器环境和 Web 应用程序。由于配置信息是以纯文本形式存储的,因此可以在没有本地管理工具帮助的情况下应用新设置。此“零本地管理”哲学也扩展到了 ASP.NET 框架应用程序的部署。只需将必要的文件复制到服务器,即可将 ASP.NET 框架应用程序部署到服务器。不需要重新启动服务器,即使是在部署或替换运行的编译代码时。

⑥ 可缩放性和可用性。ASP.NET 在设计时考虑了可缩放性,增加了专门用于在聚集环境和多处理器环境中提高性能的功能。另外,进程受到 ASP.NET 运行库的密切监视和管理,以便当进程行为不正常(泄露、死锁)时,可就地创建新进程,以帮助保持应用程序始终可用于处理请求。

⑦ 自定义性和可扩展性。ASP.NET 随附了一个设计周到的结构,它使开发人员可以在适当的级别“插入”代码。实际上,可以用自己编写的自定义组件扩展或替换 ASP.NET 运行库的任何子组件。实现自定义身份验证或状态服务一直没有变得更容易。

⑧ 安全性。借助内置的 Windows 身份验证和基于每个应用程序的配置,可以保证应用程序是安全的。

ASP .NET 语言的缺点如下。

① 数据库的连接复杂。

② 不具有跨平台性,只支持 Windows 平台。

③ 在内存使用和执行时间方面耗费非常大。

第 2 章习题答案

见资源包。

第 3 章习题答案

1. 选择题

(1) A　(2) C　(3) C　(4) D　(5) D　(6) B　(7) D　(8) A

(9) C　(10) B　(11) A　(12) C　(13) C　(14) C　(15) D

2. 简答题

(1) 在网站的页面中添加控件有哪两种方法?

① 在窗口左侧的工具箱中双击控件,则控件以默认位置、默认风格直接插入页面中。

② 将工具箱中的控件直接拖动到页面指定位置。

(2) 进入代码编辑窗口有哪两种方法?

① 双击控件,即进入控件编程界面。

② 在属性面板上部单击事件切换图标,选定特定事件后,双击相应事件,即可启动代码编辑窗口。

第 4 章习题答案

(略)

第 5 章习题答案

(略)

第 6 章习题答案

(略)

第 7 章习题答案

(略)

参 考 文 献

[1] 杨威. 网络工程设计与安装[M]. 北京：电子工业出版社，2003.
[2] 杨卫东. 网络系统集成与工程设计[M]. 北京：科学出版社，2005.
[3] 肖永生. 网络互联技术[M]. 北京：高等教育出版社，2006.
[4] 赵立群. 计算机网络管理与安全[M]. 北京：清华大学出版社，2008.
[5] 丛书编委会. 中小企业网站建设与管理(动态篇)[M]. 北京：电子工业出版社，2011.
[6] 刘志，佟晓，许银龙. 网站策划师成长之路[M]. 北京：机械工业出版社，2011.
[7] 丛书编委会. 中小企业网站建设与管理(静态篇)[M]. 北京：电子工业出版社，2011.
[8] 徐曦. 网页制作与网站建设完全学习手册[M]. 北京：清华大学出版社，2012.
[9] 李京文. 网站建设技术[M]. 北京：中国水利水电出版社，2012.
[10] 臧文科，胡坤融. 网站建设与管理[M]. 北京：清华大学出版社，2012.
[11] 何新起. 网站建设与网页设计从入门到精通[M]. 北京：人民邮电出版社，2013.
[12] 徐洪祥，刘书江. 网站建设与管理案例教程[M]. 北京：北京大学出版社，2013.
[13] 张晓景. 网页色彩搭配设计师必备宝典[M]. 北京：清华大学出版社，2014.
[14] 梁露. 中小企业建设与管理[M]. 北京：电子工业出版社，2014.
[15] 蔡永华. 网站建设与网页设计制作[M]. 北京：清华大学出版社，2014.
[16] 胡秀娥. 完全掌握网页设计和网站制作实用手册[M]. 北京：机械工业出版社，2014.
[17] 吴春明，邹显春. 网页制作与网站建设[M]. 重庆：西南师范大学出版社，2014.
[18] 蒋金楠. ASP.NET mvc 5 框架揭秘[M]. 北京：电子工业出版社，2014.
[19] 朱伟华. ASP.NET 程序设计案例教程[M]. 北京：清华大学出版社，2014.
[20] 张蓉. 网页制作与网站建设宝典[M]. 北京：电子工业出版社，2014.
[21] 刘玉红. 网站开发案例课堂：HTML5＋CSS3＋JavaScript 网页设计案例课堂[M]. 北京：清华大学出版社，2015.
[22] 范生万，王敏. 电子商务网站建设与管理[M]. 上海：华东师范大学出版社，2015.

参考网站：

[1] ASP.NET 官网：http://www.asp.net/.
[2] ASP.NET 贴吧：http://tieba.baidu.com/f?kw=asp.net&fr=ps0bt&ie=utf-8.
[3] CSDN ASP.NET 论坛：http://bbs.csdn.net/forums/ASPDotNET/.
[4] ASP.NET 源代码：http://down.admin5.com/net/.
[5] ASP.NET 源代码：http://www.51aspx.com/Type/16/.
[6] 百度、百度文库、搜狐、谷歌等网站.
[7] 设计网站大全 http://www.vipsheji.cn/.
[8] 中国教程网 http://bbs.jcwcn.com.
[9] 21 互联远程教育网 http://dx.21hulian.com.